21世纪高等院校教材

现代数值计算方法

(MATLAB版)

马昌凤　林伟川　编著

福建师范大学教材建设基金资助

科学出版社
北京

内 容 简 介

本书阐述了现代数值计算的基本理论和方法，包括数值计算的基本概念、解线性方程组的迭代法和直接法、插值法与最小二乘拟合、数值积分和数值微分、常微分方程的数值解法、非线性方程的迭代解法以及矩阵特征值问题的计算等．书中有丰富的例题、习题和上机实验题．本书既注重计算方法的实用性，又注意保持理论分析的严谨性，强调数值方法的思想和原理在计算机上的实现．选材恰当，系统性强，行文通俗流畅，具有较强的可读性．

本书的建议学时为72学时(其中含上机实验12学时)．适合作为信息与计算科学、数学与应用数学、计算机科学与技术以及统计学等专业本科生数值分析课程的教材或教学参考书，也可以作为其它理工科专业及工科研究生的数值分析参考用书．

图书在版编目(CIP)数据

现代数值计算方法：MATLAB版/马昌凤，林伟川编著．—北京：科学出版社，2008
(21世纪高等院校教材)
ISBN 978-7-03-022314-2
Ⅰ．现… Ⅱ．①马… ②林… Ⅲ．①数值计算-计算方法-高等学校-教材 ②数值计算-应用软件，MATLAB-高等学校-教材 Ⅳ．O241 O245
中国版本图书馆CIP数据核字(2008)第086738号

责任编辑：姚莉丽／责任校对：陈玉凤
责任印制：张克忠／封面设计：陈 敬

科学出版社 出版
北京东黄城根北街16号
邮政编码：100717
http://www.sciencep.com

明辉印刷有限公司印刷
科学出版社发行 各地新华书店经销
*
2008年6月第 一 版 开本：B5(720×1000)
2008年6月第一次印刷 印张：15
印数：1—4 000 字数：282 000

定价：25.00元(含光盘)
(如有印装质量问题，我社负责调换〈明辉〉)

前言

科学计算的兴起是 20 世纪最重要的科学进步之一. 随着计算机和计算方法的飞速发展, 科学计算已与科学理论和科学实验鼎立为现代科学的三大组成部分之一. 在各种科学和工程领域中逐步形成了计算性科学分支, 如计算物理、计算力学、计算化学、计算地震学等. 计算在生命科学、医学、系统科学、经济学以及社会科学中所起的作用也日益增大. 在气象、勘探、航空航天、交通运输、机械制造、水利建筑等许多重要工程领域中, 计算已经成为不可缺少的工具. 这些计算性的科学和工程领域, 又以计算方法作为其共性基础和联系纽带, 使得计算数学这一古老的数学科目成为现代数学中一个生机勃勃的分支, 它是数学科学中最直接与生活相联系的部分, 是理论到实际的桥梁.

计算方法也称为数值计算或数值分析, 其主要任务是构造求解科学和工程问题的计算方法, 研究算法的数学机理, 在计算机上设计和进行计算试验, 分析这些数值实验的误差, 并与相应的理论和可能的实验对比印证. 这就是数值计算方法研究的对象和任务.

本书各章节的主要算法都给出了 MATLAB 通用程序. 为了更好地配合本书的教学, 书后编写了两个附录. 附录一介绍了数值实验报告的格式和一些具体的数值实验题目及各个实验的目的要求; 附录二简明扼要地介绍了数学软件 MATLAB 的入门知识和必要的程序设计和绘图等基本技能技巧. 在使用本教材之前教师可先介绍这部分内容, 也可以安排学生在授课前自学这部分内容.

本书具有如下特点:

1. 讲述数值分析中最重要最基础的理论与方法, 它们是研究各种复杂的数值计算问题的基础和工具.

2. 全书根据给定算法采用当前最流行的数值分析软件 MATLAB 进行编程, 所给的各算法的通用程序都可以直接应用于实际计算. 所有程序都在计算机上经过调试和运行, 简洁而不乏准确.

3. 本书所给的每一通用程序之后都给出了相应的计算实例. 这不仅能帮助学生理解程序里所包含的数值分析理论知识, 而且对培养学生处理数值计算问题的能力也大有裨益.

4. 全书每章都配备了一定数量的习题, 习题分为理论分析题和上机实验题, 以加强学生对所学知识的理解和巩固.

本书建议学时为 72 学时 (其中含上机实验 12 学时). 如果讲授学时少于 60,

可对第 8 章内容进行适当取舍. 与本书配套出版的教学课件含 30 个 PDF 课件和教材中所列算法的全部 MATLAB 源程序, 可供学生和任课教师参考使用.

本书的编写和出版得到了福建师范大学教材建设基金资助项目和国家自然科学基金项目 (编号: 10661005) 的部分资助, 在此作者表示由衷的感谢. 由于编者水平有限, 缺点和错误在所难免, 恳请专家和读者批评指正.

编 者

2008 年 3 月于福州长安山

目 录

第 1 章　数值计算的基本概念

1.1　数值计算的研究对象和内容

数值计算是数学中关于计算的一门学问, 它研究如何借助于计算工具求得数学问题的数值解答. 这里的数学问题仅限于数值问题, 即给出一组数值型的数据 (通常是一些实数, 称为初始数据), 去求另一组数值型数据, 问题的本身反映了这两组数据之间的某种确定关系. 如函数的计算、方程的求根都是数值问题的典型例子.

数值计算的历史源远流长, 自有数学以来就有关于数值计算方面的研究. 古代巴比伦人在公元前 2000 年左右就有了关于二次方程求解的研究, 我国古代数学家刘徽利用割圆术求得圆周率的近似值, 而后祖冲之求得圆周率的高精度的值都是数值计算方面的杰出成就. 数值计算的理论与方法是在解决数值问题的长期实践过程中逐步形成和发展起来的. 但在电子计算机出现以前, 它的理论与方法发展十分缓慢, 甚至长期停滞不前. 由于受到计算工具的限制, 无法进行大量的复杂的计算.

科学技术的发展与进步提出了越来越多的复杂的数值计算问题, 这些问题的圆满解决已远非人工手算所能胜任, 必须依靠电子计算机快速准确的数据处理能力. 这种用计算机处理数值问题的方法, 称为科学计算. 今天, 科学计算的应用范围非常广泛, 天气预报、工程设计、流体计算、经济规划和预测以及国防尖端的一些科研项目, 如核武器的研制、导弹和火箭的发射等, 始终是科学计算最为活跃的领域.

现代数值计算的理论与方法是与计算机技术的发展与进步一脉相承的. 无论计算机在数据处理、信息加工等方面取得了多么辉煌的成就, 科学计算始终是计算机应用的一个重要方面, 而数值计算的理论与方法是计算机进行科学计算的依据. 它不但为科学计算提供了可靠的理论基础, 并且提供了大量行之有效的数值问题的算法.

由于计算机对数值计算这门学科的推动和影响,使数值计算的重点转移到使用计算机编程算题的方面上来. 现代的数值计算理论与方法主要是面对计算机的. 研究与寻求适合在计算机上求解各种数值问题的算法是数值计算这门学科的主要内容.

1.2　数值算法的基本概念

粗略地说, 数值算法就是求解数值问题的计算步骤. 由一些基本运算及运算顺序的规定构成的一个 (数值) 问题完整的求解方案称为 (数值) 算法.

计算机的运算速度极高, 可以承担大运算量的工作, 这是否意味着人们可以对计算机上的算法随意选择呢?

我们知道, 在线性代数中, 克拉默法则原则上可用来求解线性方程组. 用这种方法求解一个 n 阶方程组, 要计算 $n+1$ 个 n 阶行列式的值, 这意味着总共需要做 $A_n = n!(n-1)(n+1)$ 次乘法. 当 n 充分大时, 这个计算量是相当惊人的. 譬如: 对于一个 20 阶的方程组, 大约需要做 $A_{20} \approx 10^{21}$ 次乘法, 现在假设这项计算用每秒十亿次 (10^9) 的计算机去做, 每年只能完成大约 $3.15 \times 10^{16}(365 \times 24 \times 3600 \times 10^9)$, 故所需时间约为 $10^{21} \div (3.15 \times 10^16) \approx 3.2 \times 10^4$ (年), 即大约需要三万二千年才能完成. 当然, 解线性方程组我们有许多实用的算法 (参看本书的后续章节). 这个简单的例子说明, 能否正确地制定算法是科学计算成败的关键.

计算机虽然是运算速度极高的现代化计算工具, 但它本质上仅能完成一系列具有一定位数的基本的算术运算和逻辑运算. 故而在进行数值计算时, 首先要将各种类型的数值问题转化为一系列计算机能够执行的基本运算.

通常的数值问题是在实数范围内提出的, 而计算机所能表示的数仅仅是有限位小数, 误差不可避免. 这些误差对计算结果的影响是需要考虑的. 如果给出一种算法, 在计算机上运行时, 误差在成千上万次的运算过程中得不到控制, 初始数据的误差, 由中间结果的舍入产生的误差, 这些误差在计算过程中的累计越来越大, 以致淹没了真值, 那么这样的计算结果将变得毫无意义. 相应地, 我们称这种算法是不可靠的, 或者数值不稳定的. 现在的计算机无论在运算速度上还是在存储能力上都是传统计算工具所无法比拟的. 但即使这样, 我们在设计算法时, 也必须对算法的运算次数和存储量大小给予足够的重视. 实际中存在大量这样的问题, 由于所提供的解决这些问题的算法的运算量大得惊人, 即使利用最尖端的计算机也无法在有效时间内求得问题的答案.

那么, 一个好的算法一般应该具备什么特征呢? (1) 必须结构简单, 易于计算机实现; (2) 理论上必须保证方法的收敛性和数值稳定性; (3) 计算效率必须要高, 即计算速度快且节省存储量; (4) 必须经过数值实验检验, 证明行之有效.

1.3 误差的基本理论

1.3.1 误差的来源

误差是描述数值计算中近似值精确程度的一个基本概念, 在数值计算中十分重要, 误差按来源可分为模型误差、观测误差、截断误差和舍入误差四种.

1. *模型误差*

数学模型通常是由实际问题抽象得到的, 一般带有误差. 这种误差称为模型误

差.

2. 观测误差

数学模型中包含的一些物理参数通常是通过观测和实验得到的, 难免带有误差, 这种误差称为观测误差.

3. 截断误差

求解数学模型所用的数值方法通常是一种近似方法, 这种因方法产生的误差称为截断误差或方法误差. 例如, 利用 $\ln(x+1)$ 的 Taylor 公式:

$$\ln(x+1)=x-\frac{1}{2}x^2+\frac{1}{3}x^3-\frac{1}{4}x^4+\cdots+(-1)^{n-1}\frac{1}{n}x^n+\cdots$$

实际计算时只能截取有限项代数和计算, 如取前 5 项有:

$$\ln 2\approx 1-\frac{1}{2}+\frac{1}{3}-\frac{1}{4}+\frac{1}{5}.$$

这时产生误差 (记作 R_5)

$$R_5=-\frac{1}{6}+\frac{1}{7}-\frac{1}{8}+\frac{1}{9}-\cdots.$$

4. 舍入误差

由于计算机只能对有限位数进行运算, 在运算中像 e, $\sqrt{2}$, 1/3 等都要按舍入原则保留有限位, 这时产生的误差称为舍入误差或计算误差. 关于舍入误差, 以后我们还要讨论.

在数值分析中, 我们总假定数学模型是准确的, 因而不考虑模型误差和观测误差, 主要研究截断误差和舍入误差对计算结果的影响.

1.3.2 绝对误差和相对误差

1. 绝对误差

给一实数 x, 它的近似值为 x^*, x^*-x 反映了近似值和精确值差异的大小, 因此称

$$\varepsilon(x)=x^*-x$$

为近似数 x^* 的绝对误差. 由于精确值往往是无法知道的, 因此近似数的绝对误差也无法得到, 但有时却能估计出 $\varepsilon(x)$ 的绝对值的一个上限. 如果存在一个正数 η, 使得

$$|\varepsilon(x)|\leqslant\eta,$$

则称 η 为 x^* 的绝对误差限 (或误差界). 此时

$$x-\eta\leqslant x^*\leqslant x+\eta.$$

通常将上式简记为 $x^*=x\pm\eta$.

2. 相对误差

绝对误差通常不能完全反映近似数的精确程度, 它还依赖于此数本身的大小, 因此有必要引进相对误差的概念. 近似数 x^* 的相对误差定义为

$$\varepsilon_r(x)=\frac{\varepsilon(x)}{x}=\frac{x^*-x}{x}.$$

由于 x 未知, 实际使用时总是将 x^* 的相对误差取为

$$\varepsilon_r(x)=\frac{\varepsilon(x)}{x^*}=\frac{x^*-x}{x^*}.$$

和绝对误差的情况一样, 引进相对误差限 (或相对误差界) 的概念. 如果存在一个正数 δ, 使得

$$|\varepsilon_r(x)|\leqslant\delta,$$

则称 δ 为 x^* 的相对误差限.

根据上述定义可知, 当 $|x^*-x|\leqslant 1$ cm 时, 测量 10 m 物体时的相对误差为

$$|\varepsilon_r(x)|\leqslant\frac{1}{1000}=0.1\%.$$

而测量 100 m 物体时的相对误差为

$$|\varepsilon_r(x)|\leqslant\frac{1}{10000}=0.01\%.$$

可见后者的测量结果要比前者精确. 所以, 在分析误差时, 相对误差更能刻画误差的特性.

例 1.1 设 $x^*=4.32$ 是由精确值 x 经过四舍五入得到的近似值, 求 x^* 的绝对误差限和相对误差限.

解 由已知可得: $4.315\leqslant x<4.325$, 所以

$$-0.005\leqslant x-x^*<0.005.$$

因此, 绝对误差限为 $\eta=0.005$, 相对误差限为 $\delta=0.005\div 4.32\approx 0.12\%$.

3. 向量的误差

实际问题中给出的数据 (初始数据) 往往不是一个孤立的数, 有时给出的是一组相关联的实数 $x_1,x_2,\cdots,x_n$, 为了便于统一处理, 通常将它们看成 n 维欧氏空间 $\mathbf{R}^n$ 中的向量或点, 记为 $x=(x_1,x_2,\cdots,x_n)^{\mathrm{T}}$. 设 x_i^* 是 $x_i(1\leqslant i\leqslant n)$ 的近似数, $x^*=(x_1^*,x_2^*,\cdots,x_n^*)^{\mathrm{T}}$ 是 x 的近似向量, 反映这两组数据的整体误差可用 x^*-x 的欧氏范数

$$\|x^*-x\|=\Big(\sum_{i=1}^{n}(x_i^*-x_i)^2\Big)^{\frac{1}{2}}$$

来表示, 它实际上是向量 x^* 与 x 在 $\mathbf{R}^n$ 上的欧氏距离. 用 $\|x^*-x\|$ 的值表示 x^* 的绝对误差, 类似地用 $\|x^*-x\|/\|x\|$ 表示 x^* 的相对误差. 容易发现, 当 $n=1$ 时, 就化成前述通常的单个数的误差了.

1.3.3 近似数的有效数字

为了能给出一种数的表示法, 使之既能表示其大小, 又能表示其精确程度, 于是需要引进有效数字的概念. 在实际计算中, 当准确值 x 有很多位数时, 我们通常按四舍五入得到 x 的近似值 x^*. 例如无理数

$$\pi = 3.1415926535897\cdots,$$

若按四舍五入原则分别取二位和四位小数时, 则得

$$\pi \approx 3.14, \quad \pi \approx 3.1416.$$

不管取几位得到的近似数, 其绝对误差不会超过末位数的半个单位, 即

$$|\pi - 3.14| \leqslant \frac{1}{2}\times 10^{-2}, \quad |\pi - 3.1416| \leqslant \frac{1}{2}\times 10^{-4}.$$

定义 1.1 设数 x^* 是数 x 的近似值, 如果 x^* 的绝对误差限是它的某一数位的半个单位, 并且从 x^* 左起第一个非零数字到该数位共有 n 位, 则称这 n 个数字为 x^* 的有效数字, 也称用 x^* 近似 x 时具有 n 位有效数字.

例 1.2 已知下列近似数

$$a = 24.1357, \quad b = -0.250, \quad c = 2, \quad d = 0.00016$$

的绝对误差限都是 0.0005, 问它们具有几位有效数字?

解 由于 0.0005 是小数点后第 3 数位的半个单位, 所以 a 有 5 位有效数字 2、4、1、3、5, b 有 3 位有效数字 2、5、0, c 有 1 位有效数字 2, d 没有有效数字.

一般地, 任何一个实数 x 经过四舍五入后得到的近似值 x^* 都可以写成如下标准形式

$$x^* = \pm 0.a_1a_2\cdots a_n \times 10^m. \tag{1.1}$$

所以, 当其绝对误差限满足

$$|x^* - x| \leqslant \frac{1}{2}\times 10^{m-n}$$

时, 则称近似值 x^* 具有 n 位有效数字, 其中 m 为整数, a_1 是 1 到 9 中的某个数字, $a_2,\cdots,a_n$ 是 0 到 9 中的数字.

根据上述有效数字的定义, 不难验证 π 的近似值 3.1416 具有 5 位有效数字. 事实上, $3.1416 = 0.31416 \times 10^1$, 这里 $m = 1$, $n = 5$, 由于

$$|\pi - 3.1416| = 0.0000073465 \cdots < \frac{1}{2} \times 10^{-4},$$

所以它具有 5 位有效数字.

有效数字与绝对误差、相对误差有如下关系:

定理 1.1 (1) *若有数 x 的近似值 $x^* = \pm 0.a_1 a_2 \cdots a_n \times 10^m$ 有 n 位有效数字, 则此近似值 x^* 的绝对误差限为*

$$|x^* - x| \leqslant \frac{1}{2} \times 10^{m-n}. \tag{1.2}$$

(2) *若近似数 x^* 有 n 位有效数字, 则其相对误差限满足*

$$|\varepsilon_r(x)| \leqslant \frac{1}{2a_1} \times 10^{-(n-1)}. \tag{1.3}$$

反之, 若近似数 x^* 的相对误差限满足

$$|\varepsilon_r(x)| \leqslant \frac{1}{2(a_1+1)} \times 10^{-(n-1)}, \tag{1.4}$$

则 x^* 至少有 n 位有效数字.

证明 (1) 结论显然.

(2) 由 (1.1) 式可知

$$a_1 \times 10^{m-1} \leqslant |x^*| \leqslant (a_1 + 1) \times 10^{m-1},$$

故

$$|\varepsilon_r(x)| = \frac{|x^* - x|}{|x^*|} \leqslant \frac{\frac{1}{2} \times 10^{m-n}}{a_1 \times 10^{m-1}} = \frac{1}{2a_1} \times 10^{-(n-1)}.$$

反之, 由

$$|x^* - x| = |x^*| \cdot |\varepsilon_r(x)| \leqslant (a_1 + 1) \times 10^{m-1} \cdot \frac{1}{2(a_1+1)} \times 10^{-(n-1)} = \frac{1}{2} \times 10^{m-n}$$

知, x^* 至少有 n 位有效数字. □

由此可见, 当 m 一定时, n 越大 (即有效数字位数越多), 其绝对误差限越小.

例 1.3 *为使 $\sqrt{26}$ 的近似值的相对误差小于 0.1%, 问至少应取几位有效数字?*

解 $\sqrt{26}$ 的近似值的首位非零数字是 $a_1 = 5$. 假设应取 n 位有效数字, 则由 (1.3) 式有

$$|\varepsilon_r(x)| \leqslant \frac{1}{2\times 5} \times 10^{-(n-1)} < 0.1\%,$$

解之得 $n > 3$, 故取 $n = 4$ 即可满足要求. 也就是说, 只要 $\sqrt{26}$ 的近似值具有 4 位有效数字, 就能保证 $\sqrt{26} \approx 5.099$ 的相对误差小于 0.1%.

例 1.4 已知近似数 x^* 的相对误差界为 0.0002, 问 x^* 至少有几位有效数字?

解 由于 x^* 首位数未知, 但必有 $1 \leqslant a_1 \leqslant 9$, 则由 (1.4) 式有

$$|\varepsilon_r(x)| \leqslant \frac{1}{2\times(a_1+1)} \times 10^{-(n-1)} < 0.0002,$$

得

$$10^{n-1} \geqslant \frac{1}{4\times(a_1+1)} \times 10^4 \geqslant \frac{1}{4\times(9+1)} \times 10^4 = \frac{1}{4} \times 10^3,$$

解之得 $n \geqslant 3.3979$, 故取 $n = 4$.

1.4 数值算法设计的若干原则

为了减少舍入误差的影响, 设计算法时应遵循如下的一些原则.

1. 避免两个相近的数相减

如果 x^*, y^* 分别是 x, y 的近似值, 则 $z^* = x^* - y^*$ 是 $z = x - y$ 的近似值, 此时有

$$|\varepsilon_r(z)| = \frac{|z^* - z|}{|z^*|} \leqslant \left|\frac{x^*}{x^* - y^*}\right| \cdot |\varepsilon_r(x)| + \left|\frac{y^*}{x^* - y^*}\right| \cdot |\varepsilon_r(y)|,$$

可见, 当 x^* 与 y^* 很接近时, z^* 的相对误差有可能很大. 例如, 当 $x = 5000$ 时, 计算

$$\sqrt{x+1} - \sqrt{x}$$

的值, 若取 4 位有效数字计算

$$\sqrt{x+1} - \sqrt{x} = \sqrt{5001} - \sqrt{5000} = 71.72 - 71.71 = 0.01.$$

这个结果只有一位有效数字, 损失了三位有效数字, 从而绝对误差和相对误差都变得很大, 严重影响了计算精度. 但如果将公式改变为

$$\sqrt{x+1} - \sqrt{x} = \frac{1}{\sqrt{x+1} + \sqrt{x}} = \frac{1}{\sqrt{5001} + \sqrt{5000}} \approx 0.006972.$$

它仍然有四位有效数字, 可见改变计算公式可以避免两个相近数相减而引起的有效数字的损失, 从而得到比较精确的计算结果.

因此, 在数值计算中, 如果遇到两个相近的数相减运算, 应考虑改变一下算法以避免两数相减.

2. 避免绝对值太小的数作除数

由于除数很小, 将导致商很大, 有可能出现"溢出"现象. 另外, 设 x^*, y^* 分别是 x, y 的近似值, 则 $z^* = x^* \div y^*$ 是 $z = x \div y$ 的近似值, 此时, z 的绝对误差满足

$$|\varepsilon(z)| = |z^* - z| = \left|\frac{(x^* - x)y + x(y - y^*)}{y^* y}\right| \approx \frac{|x^*| \cdot |\varepsilon(y)| + |y^*| \cdot |\varepsilon(x)|}{(y^*)^2}.$$

由此可见, 若除数太小, 则可能导致商的绝对误差很大.

3. 防止大数"吃掉"小数

因为计算机上只能采用有限位数计算, 若参加运算的数量级差很大, 在它们的加、减运算中, 绝对值很小的数往往被绝对值较大的数"吃掉", 造成计算结果失真.

例 1.5　*求方程*

$$x^2 - (10^9 + 1)x + 10^9 = 0$$

的根.

解　显然, 方程的两个根为: $x_1 = 10^9$, $x_2 = 1$. 如果用 8 位数字的计算机计算, 使用二次方程的求根公式

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

得到

$$\begin{aligned} -b &= 10^9 + 1 = \underbrace{0.10000000}_{8\text{ 位}} \times 10^{10} + \underbrace{0.10000000}_{8\text{ 位}} \times 10^1 \\ &= \underbrace{0.10000000}_{8\text{ 位}} \times 10^{10} + \underbrace{0.00000000}_{8\text{ 位}} \times 10^{10} \\ &\triangleq 0.10000000 \times 10^{10} \quad (\text{第 9,10 位舍去}) \\ &= 10^9 \quad (\triangleq \text{ 表示机器中的相等}), \end{aligned}$$

那么有

$$\sqrt{b^2 - 4ac} = \sqrt{(10^9 + 1)^2 - 4 \times 1 \times 10^9} = \sqrt{(10^9 - 1)^2} \triangleq 10^9,$$

所以

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} = \frac{10^9 + 10^9}{2 \times 1} = 10^9, \quad x_2 = \frac{10^9 - 10^9}{2 \times 1} = 0.$$

实际上, x_2 应等于 1. 可以看出 x_2 的误差太大, 原因是在做加减运算过程中要"对阶", 因而"小数" 1 在对阶过程中, 被"大数" 10^9 吃掉了. 而从上述计算可以看出, x_1 是可靠的, 故可利用根与系数的关系 $x_1 \cdot x_2 = c/a$ 来求 x_2:

$$x_2 = \frac{c}{ax_1} = \frac{10^9}{1 \times 10^9} = 1.$$

此方法是可靠的.

4. *尽量简化计算步骤以减少运算次数*

同样一个问题, 如果能减少运算次数, 不但可以节省计算时间, 还可以减少舍入误差的传播. 这是数值计算中必须遵循的原则, 也是数值分析要研究的重要内容. 例如, 计算多项式

$$P(x) = a_0x^n + a_1x^{n-1} + \cdots + a_{n-1}x + a_n$$

的值. 若直接计算 $a_{n-k}x^k (k = 0, 1, \cdots, n)$, 再逐项相加; 一共需做

$$1 + 2 + \cdots + (n-1) + n = \frac{n(n+1)}{2}$$

次乘法和 n 次加法. 而若采用

$$P(x) = (((a_0x + a_1)x + a_2)x + \cdots + a_{n-1})x + a_n$$

所谓的秦九韶算法, 则只需要 n 次乘法和 n 次加法即可. 又如, 计算 x^{63} 的值. 若将 x 的值逐个相乘, 则需做 62 次乘法, 但如果写成

$$x^{63} = x \cdot x^2 \cdot x^4 \cdot x^8 \cdot x^{16} \cdot x^{32},$$

只要 10 次乘法运算就可以了.

5. *应采用数值稳定性好的算法*

在计算过程中, 由于原始数据本身就具有误差, 每次运算有可能产生舍入误差. 误差的传播和累积很可能会淹没真解, 使计算结果变得根本不可靠. 先看下面的例子.

例 1.6 在四位十进制计算机上计算 8 个积分:

$$I_n = \int_0^1 x^n e^{x-1} dx, \quad n = 0, 1, \cdots, 7.$$

解 利用分部积分公式可得递推关系: $I_n = 1 - nI_{n-1}$. 注意到 $I_0 = 1 - e^{-1} \approx 0.6321$ 及

$$I_n = \int_0^1 x^n e^{-(1-x)} dx < \int_0^1 x^n dx = \frac{1}{n+1} \to 0 \ (n \to \infty),$$

可得两种算法：

算法 (a) 令 $I_0 = 0.6321$, 再算 $I_n = 1 - nI_{n-1}$, $n = 1, 2, \cdots, 7$;

算法 (b) 令 $I_{11} = 0$, 再算 $I_{n-1} = (1 - I_n)/n$, $n = 11, 10, \cdots, 1$.

按算法 (a) 得 8 个积分的近似值为

$$0.6321,\ 0.3679, 0.2642, 0.2074, 0.1704, 0.1480, 0.1120, 0.2160.$$

按算法 (b) 得 8 个积分的近似值为

$$0.6321,\ 0.3679, 0.2642, 0.2073, 0.1709, 0.1455, 0.1268, 0.1124.$$

算法 (b) 中的诸结果均准确到 4 位小数, 而算法 (a) 中的 I_7 没有一位数字是准确的. □

可靠的算法, 各步误差不应对计算结果产生过大的影响, 即具有稳定性. 研究算法是否稳定, 理应考察每一步的误差对算法的影响, 但这相当困难且繁琐. 为简单计, 通常只考虑某一步 (如运算开始时) 误差的影响. 这实质上是把算法稳定性的研究, 转化为初始数据误差对算法影响的分析. 从某种意义上来说, 这种做法是合理的. 可以设想, 一步误差影响大, 多步误差影响更大; 一步误差影响逐步削弱, 多步误差影响也削弱. 因此简化研究得出的结论具有指导意义.

下面用此法分析例 1.6 中的两种算法的稳定性. 对算法 (a), 设 $\bar{I}_0 \approx I_0$ 有误差, 此后计算无误差, 则

$$\begin{cases} I_n = 1 - nI_{n-1}, \\ \bar{I}_n = 1 - n\bar{I}_{n-1}, \end{cases} \quad n = 1, 2, \cdots, 7.$$

两式相减得

$$I_n - \bar{I}_n = (-n)(I_{n-1} - \bar{I}_{n-1}), \quad n = 1, 2, \cdots, 7.$$

由此递推, 得

$$I_7 - \bar{I}_7 = -7!(I_0 - \bar{I}_0) = -5040(I_0 - \bar{I}_0).$$

同理, 对算法 (b), 设 $\bar{I}_{11} \approx I_{11}$ 有误差, 此后计算无误差, 则有

$$I_0 - \bar{I}_0 = -(I_{11} - \bar{I}_{11})/39916800 \approx -2.5052 \times 10^{-8}(I_{11} - \bar{I}_{11}).$$

由此可见, 按算法 (a), 最后结果误差是初始误差的 5040 倍, 而按算法 (b), 最后结果误差只是初始误差的 39916800 分之一. 这说明算法 (b) 的稳定性较好.

习 题 1

1.1 下列近似数都是通过四舍五入得到的, 指出它们的绝对误差限、相对误差限和有效数字位数.

(1) 23000, (2) 0.00230, (3) 2300.00, (4) 2.30×10^4.

1.2 下列各数都是经过四舍五入得到的近似值, 求各数的绝对误差限、相对误差限和有效数字的位数.

(1) 3580, (2) 0.0476, (3) 30.120, (4) 0.3012×10^{-5}.

1.3 已知 $\pi = 3.141592654\cdots$, 问:

(1) 若其近似值取 5 位有效数字, 则该近似值是多少? 其误差限是多少?

(2) 若其近似值精确到小数点后面 4 位, 则该近似值是什么? 其误差限是什么?

(3) 若其近似值的绝对误差限位 0.5×10^{-5}, 则该近似值是什么?

1.4 已知近似数 x^* 的相对误差限为 0.01%, 问 x^* 至少有几位有效数字?

1.5 已知近似数 x^* 有 3 位有效数字, 试求其相对误差限.

1.6 要使 $\sqrt{6}$ 的近似值的相对误差限小于 0.1%, 需取几位有效数字?

1.7 已知近似数 x^* 的相对误差限为 0.3%, 问 x^* 至少有几位有效数字?

1.8 求方程 $x^2-56x+1=0$ 的两个根, 使之至少具有四位有效数字 (已知 $\sqrt{783} \approx 27.982$).

1.9 为了使无理数 e 的近似值的相对误差不超过 0.01%, 问应取几位有效数字?

1.10 怎样计算下列个体才能使得结果比较精确?

(1) $\sin(x+y)-\sin x$, 其中 $|y|$ 充分小;

(2) $1-\cos 1^\circ$;

(3) $\ln\left(\sqrt{10^{10}+1}\right)-10^5$).

1.11 已知 $|x| \ll 1$, 下列计算 y 的公式哪一个比较准确?

(1) (A) $y = \dfrac{2\sin^2 x}{x}$, (B) $y = \dfrac{1-\cos 2x}{x}$;

(2) (A) $y = \ln\dfrac{|x|}{1+\sqrt{1-x^2}}$, (B) $y = \ln\dfrac{1-\sqrt{1-x^2}}{|x|}$.

1.12 已知积分 $I_n = \displaystyle\int_0^1 \frac{x^n}{x+4}\mathrm{d}x$ 具有递推关系:

$$I_n = \frac{1}{n} - 4I_{n-1}, \quad n = 1, 2, \cdots,$$

试在 4 位十进制计算机上利用下面两种算法计算积分 $I_0, I_1, \cdots, I_7$:

(1) 算法 1: 令 $I_0 = 0.2231(\approx \ln 1.25)$, 计算 $I_n = \dfrac{1}{n} - 4I_{n-1}$, $n = 1, 2, \cdots, 7$;

(2) 算法 2: 令 $I_{11} = 0$, 计算 $I_{n-1} = \dfrac{1}{4n}(1-nI_n)$, $n = 11, 10, \cdots, 1$.

哪种算法准确? 原因是什么?

第 2 章　解线性方程组的迭代法

在工程计算和科学研究中, 经常会遇到求解 n 阶线性代数方程组的问题. 这种方程组具有如下形式:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2, \\ \cdots\cdots\cdots\cdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n. \end{cases} \tag{2.1}$$

为了讨论方便, 仅考虑实系数的方程组, 即系数 a_{ij} 和常数项 b_i 均为实数, x_i 称为未知数. 记 $A = (a_{ij})_{n\times n}$, $b = (b_1, b_2, \cdots, b_n)^{\mathrm{T}}$, 则可将 (2.1) 写成矩阵形式

$$Ax = b. \tag{2.2}$$

线性方程组的解法大致分为精确解法 (直接法) 和迭代解法 (间接法) 两大类. 本章首先介绍迭代解法, 这类算法的一个突出优点是算法简单, 因而编制程序比较容易. 计算实践表明, 迭代法对于求解大型稀疏方程组是十分有效的, 因为它可以保持系数矩阵稀疏的优点, 从而节省大量的存储量和计算量.

本章的目的就是介绍求解线性方程组 (2.1) 的迭代解法. 主要介绍雅可比 (Jacobi) 迭代法, 高斯–赛德尔 (Gauss-Seidel) 迭代法, 逐次超松弛 (SOR) 迭代法, 并讨论各类迭代算法的收敛性.

2.1　迭代法的一般理论

2.1.1　向量范数和矩阵范数

首先介绍向量范数和矩阵范数, 它们是迭代法理论的基础.

1. 向量范数

定义 2.1　设 $x = (x_1, x_2, \cdots, x_n)^{\mathrm{T}}$, $y = (y_1, y_2, \cdots, y_n)^{\mathrm{T}}$, 称

$$(x, y) = x^{\mathrm{T}}y = \sum_{i=1}^{n} x_i y_i$$

为向量 x 和 y 的内积.

向量内积具有如下性质：

(1) $(x,x)\geqslant 0$, 当且仅当 $x=0$ 时, $(x,x)=0$；　(非负性)

(2) $(x,y)=(y,x)$；　(对称性)

(3) $(\alpha x,y)=\alpha(x,y)$, 其中 α 为某一实数；　(齐次性)

(4) $(x+y,z)=(x,z)+(y,z)$.　(可加性)

定义 2.2　若对 $x,y\in \mathbf{R}^n$, 有

(1) $\|x\|\geqslant 0$, 且 $\|x\|=0$ 当且仅当 $x=0$;

(2) $\|\alpha x\|=|\alpha|\cdot\|x\|$;

(3) $\|x+y\|\leqslant\|x\|+\|y\|$,

则称 $\|x\|$ 为向量 x 的范数. 定义了范数的线性空间称为赋范线性空间.

常用的向量范数有

(i) $\|x\|_1=\sum\limits_{i=1}^{n}|x_i|$,　(1-范数)

(ii) $\|x\|_2=\sqrt{(x,x)}=\left(\sum\limits_{i=1}^{n}x_i^2\right)^{1/2}$,　(2-范数)

(iii) $\|x\|_\infty=\max\limits_{1\leqslant i\leqslant n}|x_i|$.　(无穷范数)

不难验证, 上述三种范数均满足范数的定义.

2. 矩阵范数

将 $m\times n$ 矩阵 A 看作线性空间 $\mathbf{R}^{m\times n}$ 中的元素, 则完全可以按照定义 2.2 的方式引入矩阵的范数. 其中最常用的是与向量 2-范数相对应的范数

$$\|A\|_F=\left(\sum_{i=1}^{m}\sum_{j=1}^{n}a_{ij}^2\right)^{1/2}$$

称为矩阵 A 的 F-范数.

关于矩阵范数与向量范数之间的关系, 引入相容性的概念.

定义 2.3　对于给定 $\mathbf{R}^n$ 上的一种范数 $\|x\|$ 和 $\mathbf{R}^{m\times n}$ 上的一种范数 $\|A\|$, 若有

$$\|Ax\|\leqslant\|A\|\cdot\|x\|,\quad \forall x\in\mathbf{R}^n,\ A\in\mathbf{R}^{m\times n},$$

则称上述矩阵范数和向量范数是相容的.

这样, 可以利用相容性定义矩阵的范数, 定义

$$\|A\|=\max_{x\neq 0}\frac{\|Ax\|}{\|x\|}=\max_{\|x\|=1}\|Ax\| \tag{2.3}$$

为矩阵 A 的范数, 称为相容性范数.

按 (2.3) 可以求出矩阵常用的三种相容性范数:

(1) $\|A\|_1 = \max\limits_{1\leqslant j\leqslant n} \sum\limits_{i=1}^{m} |a_{ij}|$, (列和范数)

(2) $\|A\|_2 = \sqrt{\lambda_{\max}(A^{\mathrm{T}}A)}$, (谱范数)

(3) $\|A\|_\infty = \max\limits_{1\leqslant i\leqslant m} \sum\limits_{j=1}^{n} |a_{ij}|$, (行和范数)

其中 $\lambda_{\max}(A^{\mathrm{T}}A)$ 表示矩阵 $A^{\mathrm{T}}A$ 的最大特征值.

3. 谱半径

定义 2.4 设 $A \in \mathbf{R}^{n\times n}$, 其特征值为 $\lambda_1, \lambda_2, \cdots, \lambda_n$, 则称

$$\rho(A) = \max_{1\leqslant i\leqslant n} |\lambda_i|$$

为矩阵 A 的谱半径.

由上述定义, $\|A\|_2$ 可定义为

$$\|A\|_2 = \sqrt{\rho(A^{\mathrm{T}}A)}.$$

特别地, 当 A 为对称矩阵时

$$\|A\|_2 = \rho(A).$$

对于一般情况, 有如下定理.

定理 2.1 设 $A \in \mathbf{R}^{n\times n}$, 则 A 的谱半径不超过 A 的任何范数, 即

$$\rho(A) \leqslant \|A\|.$$

证 设 λ 是 A 的一个特征值, 且 A 的谱半径 $\rho(A) = |\lambda|$, u 是对应于 λ 的特征向量, 即 $Au = \lambda u$, 所以对任何一种向量范数, 有 $\|Au\| = |\lambda| \cdot \|u\|$, 故有

$$\rho(A) = |\lambda| = \frac{\|Au\|}{\|u\|} \leqslant \max_{\|x\|\neq 0} \frac{\|Ax\|}{\|x\|} = \|A\|.$$

□

定理 2.2 对任意的 $A \in \mathbf{R}^{n\times n}$ 和任意正数 ε, 一定存在某种矩阵范数 $\|A\|_\alpha$, 使得

$$\|A\|_\alpha \leqslant \rho(A) + \varepsilon.$$

(证明略.)

关于矩阵范数, 还有下述结论:

定理 2.3 若 $\|A\| < 1$, 则矩阵 $I - A$ 非奇异, 且满足

$$\|(I-A)^{-1}\| \leqslant \frac{1}{1-\|A\|}.$$

证 用反证法. 设 $\det(I - A) = 0$, 则方程 $(I - A)x = 0$ 有非零解, 即存在 $x_0 \in \mathbf{R}^n$, $x_0 \neq 0$, 使得 $(I - A)x_0 = 0$. 故

$$\|A\| = \max_{\|x\|\neq 0} \frac{\|Ax\|}{\|x\|} \geqslant \frac{\|Ax_0\|}{\|x_0\|} = 1.$$

这与 $\|A\| < 1$ 矛盾.

进一步, 由于 $(I - A)(I - A)^{-1} = I$, 则 $(I - A)^{-1} = I + A(I - A)^{-1}$, 从而

$$\|(I-A)^{-1}\| \leqslant \|I\| + \|A\| \cdot \|(I-A)^{-1}\|.$$

将上式整理即得要证的结论. □

2.1.2 迭代格式的构造

首先将方程 (2.2) 的系数矩阵 A 分裂为

$$A = N - P, \tag{2.4}$$

这里要求 N 非奇异, 于是方程 (2.2) 等价地可写成

$$x = N^{-1}Px + N^{-1}b.$$

构造迭代公式如下

$$x^{(k+1)} = Mx^{(k)} + f, \tag{2.5}$$

其中 $M = N^{-1}P$, $f = N^{-1}b$. 称 M 为迭代格式 (2.5) 的迭代矩阵.

对于方程组 (2.2) 和迭代格式 (2.5), 若存在非奇异矩阵 Q, 使

$$M = I - QA, \quad f = Qb, \tag{2.6}$$

则称 Q 为分裂矩阵.

当任选一个解的初始近似 $x^{(0)}$ 后, 即可由 (2.5) 产生一个向量序列 $\{x^{(k)}\}$, 如果它是收敛的, 即

$$\lim_{k\to\infty} x^{(k)} = x^*,$$

对 (2.5) 两边取极限得

$$x^* = Mx^* + f.$$

若 (2.6) 式成立, 可得 $Ax^* = b$, 故 x^* 满足方程组 (2.2), 即迭代格式 (2.5) 和方程组 (2.2) 相容.

对迭代格式 (2.5), 定义误差向量

$$e^{(k)} = x^{(k)} - x^*,$$

则误差向量有如下的递推关系:

$$e^{(k)} = Me^{(k-1)} = M^2 e^{(k-2)} = \cdots = M^k e^{(0)}, \tag{2.7}$$

这里, $e^{(0)} = x^{(0)} - x^*$ 是解的初始近似 $x^{(0)}$ 与精确解的误差.

引进误差向量后, 迭代的收敛问题就等价于误差向量序列收敛于 0 的问题.

2.1.3 迭代的收敛性

欲使迭代格式 (2.5) 对任意的初始向量 $x^{(0)}$ 都收敛, 误差向量 $e^{(k)}$ 应对任意的初始误差 $e^{(0)}$ 都收敛于零向量. 于是迭代格式 (2.5) 对任意的初始向量都收敛的充分必要条件是

$$\lim_{k\to\infty} M^k = 0. \tag{2.8}$$

定理 2.4 *迭代格式 (2.5) 对任意的初始向量 $x^{(0)}$ 都收敛的充分必要条件是 $\rho(M) < 1$, 这里 M 是迭代矩阵, $\rho(M)$ 表示 M 的谱半径.*

证 必要性. 设对初始向量 $x^{(0)}$ 迭代格式 (2.5) 是收敛的, 那么 (2.8) 成立. 由定理 2.1, 对于任意的矩阵范数, 成立关系式

$$\rho(M) \leqslant \|M\|.$$

若 $\rho(M) < 1$ 不成立, 即 $\rho(M) \geqslant 1$, 则

$$\|M^k\| \geqslant \rho(M^k) = [\rho(M)]^k \geqslant 1,$$

这与 (2.8) 成立矛盾.

充分性. 若 $\rho(M) < 1$, 则存在一个正数 ε, 使得

$$\rho(M) + 2\varepsilon < 1.$$

根据定理 2.2, 存在一种矩阵范数 $\|M\|$, 使

$$\|M\| < \rho(M) + \varepsilon < 1 - \varepsilon.$$

故得

$$\|M^k\| \leqslant \|M\|^k < (1-\varepsilon)^k,$$

从而当 $k \to \infty$ 时, $\|M^k\| \to 0$, 即 $M^k \to 0$, 充分性得证. □

由此可见, 迭代是否收敛仅与迭代矩阵的谱半径有关, 即仅与方程组的系数矩阵和迭代格式的构造有关, 而与方程组的右端向量 b 及初始向量 $x^{(0)}$ 无关.

如果迭代格式是收敛的, 我们还可以给出近似解与准确解的误差估计.

定理 2.5 *设 M 为迭代矩阵, 若 $\|M\| = q < 1$, 则对迭代 (2.5), 有误差估计式*

$$\|x^{(k)} - x^*\| \leqslant \frac{q^k}{1-q}\|x^{(0)} - x^{(1)}\|. \tag{2.9}$$

证 由 (2.7) 有

$$\|x^{(k)} - x^*\| = \|e^{(k)}\| \leqslant \|M^k\| \cdot \|e^{(0)}\| \leqslant q^k \|e^{(0)}\|.$$

注意到 $x^* = (I - M)^{-1} f$, 于是

$$\begin{aligned}\|e^{(0)}\| &= \|x^{(0)} - x^*\| = \|x^{(0)} - (I - M)^{-1} f\| \\ &= \|(I - M)^{-1}[(I - M)x^{(0)} - f]\| \\ &= \|(I - M)^{-1}(x^{(0)} - x^{(1)})\| \\ &\leqslant \|(I - M)^{-1}\| \cdot \|x^{(0)} - x^{(1)}\|.\end{aligned}$$

因 $\|M\| < 1$, 根据定理 2.3 有

$$\|(I - M)^{-1}\| \leqslant \frac{1}{1-q},$$

于是

$$\|x^{(k)} - x^*\| \leqslant \frac{q^k}{1-q}\|x^{(0)} - x^{(1)}\|.$$

□

在理论上, 可用上述定理来估计近似解达到某一精度所需要的迭代次数, 但由于 q 不易计算, 故计算实践中很少使用.

定理 2.6 *若 $\|M\| < 1$, 则对任一初始近似 $x^{(0)}$, 由迭代格式 (2.5) 产生的向量序列 $\{x^{(k)}\}$ 收敛, 且有估计式*

$$\|x^{(k)} - x^*\| \leqslant \frac{\|M\|}{1 - \|M\|}\|x^{(k)} - x^{(k-1)}\|. \tag{2.10}$$

证 收敛性由定理 2.4 是显然的. 下证误差估计式 (2.10). 由于

$$\begin{aligned}e^{(k)} &= x^{(k)} - x^* = (Mx^{(k-1)} + f) - (Mx^* + f) \\ &= Mx^{(k-1)} - Mx^* = Mx^{(k-1)} - M(I - M)^{-1} f \\ &= M(I - M)^{-1}[(I - M)x^{(k-1)} - f] \\ &= M(I - M)^{-1}(x^{(k-1)} - x^{(k)}),\end{aligned}$$

利用定理 2.3, 对上式两边取范数即得定理的结论. □

由上述定理可知, 只要 $\|M\|$ 不很接近于 1, 则可用 $\{x^{(k)}\}$ 的相邻两项之差的范数 $\|x^{(k)}-x^{(k-1)}\|$ 来估计 $\|x^{(k)}-x^*\|$ 的大小.

2.2 雅可比迭代法

2.2.1 迭代公式及其通用程序

对于方程组 (2.1), 设 $a_{ii}\neq 0,\ i=1,\cdots,n$, 可以得到

$$a_{ii}x_i=b_i-\sum_{j=1,j\neq i}^{n}a_{ij}x_j,\quad i=1,\cdots,n,$$

即

$$x_i=\frac{1}{a_{ii}}\Big(b_i-\sum_{j=1,j\neq i}^{n}a_{ij}x_j\Big),\quad i=1,\cdots,n.$$

其相应的迭代公式为

$$x_i^{(k+1)}=\frac{1}{a_{ii}}\Big(b_i-\sum_{j=1,j\neq i}^{n}a_{ij}x_j^{(k)}\Big),\quad i=1,\cdots,n. \tag{2.11}$$

迭代公式 (2.11) 称为雅可比 (Jacobi) 迭代法.

为便于收敛性分析, 可将分量形式的迭代公式 (2.11) 改写成矩阵形式. 令

$$N=D=\mathrm{diag}(a_{11},a_{22},\cdots,a_{nn}),$$

因 $a_{ii}\neq 0,\ i=1,\cdots,n$, 故 N 非奇异. 对 A 作分裂得

$$A=(A-D)+D,$$

则方程组 $Ax=b$ 可改写为

$$Dx=(D-A)x+b,$$

因此有

$$x=D^{-1}(D-A)x+D^{-1}b.$$

相应的迭代公式为

$$x^{(k+1)}=D^{-1}(D-A)x^{(k)}+D^{-1}b, \tag{2.12}$$

简记为

$$x^{(k+1)}=B_Jx^{(k)}+f_J, \tag{2.13}$$

其中 $B_J=D^{-1}(D-A)=I-D^{-1}A$, $f_J=D^{-1}b$. 迭代公式 (2.12) 或 (2.13) 也称为 Jacobi 迭代, 同时称 (2.13) 中的 B_J 为 Jacobi 迭代矩阵.

下面给出 Jacobi 迭代法的具体算法步骤:

算法 2.1 (Jacobi 迭代法)

步 1 取初始点 $x^{(0)}$, 精度要求 ε, 最大迭代次数 N, 置 $k:=0$;

步 2 由 (2.11) 或 (2.12) 计算 $x^{(k+1)}$;

步 3 若 $\|x^{(k+1)}-x^{(k)}\|_\infty \leqslant \varepsilon$, 则停算, 输出 $x^{(k+1)}$ 作为方程组的近似解;

步 4 若 $k=N$, 则停算, 输出迭代失败信息; 否则置 $k:=k+1$, 转步 2.

根据算法 2.1, 编制 MATLAB 程序如下:

- Jacobi 迭代法 MATLAB 程序

```
%majacobi.m
function x=majacobi(A,b,x0,ep,N)
%用途: 用Jacobi迭代法解线性方程组Ax=b
%格式: x=majacobi(A,b,x0,ep,N)  A为系数矩阵, b为右端向量,
%x0为初始向量(默认零向量), ep为精度(默认1e-6),N为最大迭
%代次数(默认500次), x返回近似解向量
n=length(b);
if nargin<5,N=500;end
if nargin<4,ep=1e-6;end
if nargin<3,x0=zeros(n,1); end
x=zeros(n,1); k=0;
while k<N
  for i=1:n
    x(i)=(b(i)-A(i,[1:i-1,i+1:n])*x0([1:i-1,i+1:n]))/A(i,i);
  end
  if norm(x-x0,inf)<ep, break; end
  x0=x ;k=k+1;
end
if k==N, Warning('已达到迭代次数上限'); end
disp(['k=',num2str(k)])
```

例 2.1 用 Jacobi 迭代法程序 majacobi.m 解线性方程组:

$$\begin{pmatrix} 0.76 & -0.01 & -0.14 & -0.16 \\ -0.01 & 0.88 & -0.03 & 0.06 \\ -0.14 & -0.03 & 1.01 & -0.12 \\ -0.16 & 0.06 & -0.12 & 0.72 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 0.68 \\ 1.18 \\ 0.12 \\ 0.74 \end{pmatrix}.$$

取初始点 $x^{(0)}=(0,0,0,0)^{\mathrm{T}}$, 精度要求 $\varepsilon=10^{-6}$.

解 在 MATLAB 命令窗口执行程序 majacobi.m:

```
>>A=[0.76 -0.01 -0.14 -0.16; -0.01 0.88 -0.03 0.05;
```

```
   -0.14 -0.03 1.01 -0.12; -0.16 0.05 -0.12 0.72];
>> b=[0.68 1.18 0.12 0.74]';
>> x=majacobi(A,b)
```

得到计算结果:

```
k = 13
x =
    1.27616261026619
    1.29806392739565
    0.48904201392258
    1.30273287985933
```

2.2.2 收敛性分析

对于 Jacobi 迭代法, 我们有下面的收敛性定理.

定理 2.7 *若线性方程组 (2.1) 的系数矩阵 A 满足下列条件之一, 则 Jacobi 迭代收敛:*

(1) $\|B_J\|_\infty = \max\limits_i \sum\limits_{j=1,j\neq i}^{n} \dfrac{|a_{ij}|}{|a_{ii}|} < 1$;

(2) $\|B_J\|_1 = \max\limits_j \sum\limits_{i=1,i\neq j}^{n} \dfrac{|a_{ij}|}{|a_{jj}|} < 1$;

(3) $\|B_J^{\mathrm{T}}\|_\infty = \max\limits_i \sum\limits_{j=1,j\neq i}^{n} \dfrac{|a_{ji}|}{|a_{ii}|} < 1$.

证 (1) 对于 Jacobi 迭代,

$$\rho(B_J) \leqslant \|B_J\|_\infty = \max_i \sum_{j=1,j\neq i}^{n} \frac{|a_{ij}|}{|a_{ii}|} < 1,$$

所以 Jacobi 迭代收敛.

(2) 同理

$$\rho(B_J) \leqslant \|B_J\|_1 = \max_j \sum_{i=1,i\neq j}^{n} \frac{|a_{ij}|}{|a_{jj}|} < 1,$$

所以 Jacobi 迭代收敛.

(3) 由于

$$\begin{aligned}\rho(B_J) &= \rho(I - D^{-1}A) = \rho[(I - D^{-1}A)^{\mathrm{T}}] \\ &= \rho(I - A^{\mathrm{T}}D^{-1}) \leqslant \|I - A^{\mathrm{T}}D^{-1}\|_\infty \\ &= \max_i \sum_{j=1,j\neq i}^{n} \frac{|a_{ji}|}{|a_{ii}|} < 1,\end{aligned}$$

所以 Jacobi 迭代收敛. □

注 2.1 定理 2.7 只是雅可比迭代收敛的充分条件而非必要条件. 请仔细领会下面的例题.

例 2.2 证明用雅可比迭代法求解下列方程组是收敛的:

$$\begin{pmatrix} 3 & 0 & -2 \\ 0 & 2 & 1 \\ -2 & 1 & 2 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}=\begin{pmatrix} 2 \\ -1 \\ 3 \end{pmatrix}.$$

证 显然方程组的系数矩阵不满足定理 2.7 的条件, 因此不能使用该定理来判断迭代法的收敛性.

注意到, 雅可比迭代矩阵为

$$B_J=\begin{pmatrix} 0 & 0 & 2/3 \\ 0 & 0 & -1/2 \\ 1 & -1/2 & 0 \end{pmatrix}.$$

$$\det(\lambda I-B_J)=\begin{vmatrix} \lambda & 0 & -2/3 \\ 0 & \lambda & 1/2 \\ -1 & 1/2 & \lambda \end{vmatrix}=\lambda\left(\lambda^2-\frac{1}{4}\right)-\frac{2}{3}\lambda=0,$$

其特征根为

$$\lambda_1=0,\ \lambda_2=-\sqrt{\frac{11}{12}},\ \lambda_3=\sqrt{\frac{11}{12}}.$$

由于谱半径

$$\rho(B_J)=\sqrt{\frac{11}{12}}<1,$$

故雅可比迭代收敛. □

2.3 高斯-赛德尔迭代法

2.3.1 迭代公式及其通用程序

对 Jacobi 迭代方法作如下改变：迭代时首先用 $x^{(k)}=(x_1^{(k)},x_2^{(k)},\cdots,x_n^{(k)})^{\mathrm{T}}$ 代入 Jacobi 迭代的第一个方程求 $x_1^{(k+1)}$, 求得 $x_1^{(k+1)}$ 后, 用 $x_1^{(k+1)}$ 替换 $x_1^{(k)}$, 用 $(x_1^{(k+1)},x_2^{(k)},\cdots,x_n^{(k)})^{\mathrm{T}}$ 代入 Jacobi 迭代的第二个方程求 $x_2^{(k+1)}$, 求得 $x_2^{(k+1)}$ 后, 即可替换 $x_2^{(k)}$, 用 $(x_1^{(k+1)},x_2^{(k+1)},x_3^{(k)},\cdots,x_n^{(k)})^{\mathrm{T}}$ 代入 Jacobi 迭代的第三个方程求 $x_3^{(k+1)}$, 如此逐个替换, 直到 $x^{(k)}$ 的所有分量替换完成, 即可得到 $x^{(k+1)}$. 这种改变既可以节省存储量, 编程又十分方便, 这就是高斯-赛德尔 (Gauss-Seidel) 迭代.

对于线性代数方程组 (2.1), Gauss-Seidel 迭代的计算格式为

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\right), \quad i = 1, 2, \cdots, n. \tag{2.14}$$

为便于收敛性分析, 可将分量形式的迭代公式 (2.14) 改写成矩阵形式. 令

$$D = \begin{pmatrix} a_{11} & & & \\ & a_{22} & & \\ & & \ddots & \\ & & & a_{nn} \end{pmatrix},$$

$$L = \begin{pmatrix} 0 & & & \\ -a_{21} & 0 & & \\ \vdots & \vdots & \ddots & \\ -a_{n1} & -a_{n2} & \cdots & 0 \end{pmatrix},$$

$$U = \begin{pmatrix} 0 & -a_{12} & \cdots & -a_{1n} \\ & 0 & \cdots & -a_{2n} \\ & & \ddots & \vdots \\ & & & 0 \end{pmatrix},$$

则 $A = D - L - U$. 迭代 (2.14) 可表示为

$$Dx^{(k+1)} = Lx^{(k+1)} + Ux^{(k)} + b,$$

得 Gauss-Seidel 迭代的矩阵表示为

$$x^{(k+1)} = (D-L)^{-1}Ux^{(k)} + (D-L)^{-1}b, \tag{2.15}$$

简记为

$$x^{(k+1)} = B_S x^{(k)} + f_S, \tag{2.16}$$

其中 $B_S = (D-L)^{-1}U$, $f_S = (D-L)^{-1}b$.

下面给出 Gauss-Seidel 迭代法的具体算法步骤:

算法 2.2 (Gauss-Seidel 迭代法)

步 1 *输入矩阵 A, 右端向量 b, 初始点 $x^{(0)}$, 精度要求 ε, 最大迭代次数 N, 置 $k := 0$;*

步 2 计算

$$x_1 = \left(b_1 - \sum_{j=2}^{n} a_{1j}x_j^{(0)}\right) / a_{11},$$

$$x_i = \left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j - \sum_{j=i+1}^{n} a_{ij}x_j^{(0)}\right) / a_{ii}, \quad i = 2, \cdots, n-1,$$

$$x_n = \left(b_n - \sum_{j=1}^{n-1} a_{nj}x_j\right) / a_{nn};$$

步 3 若 $\|x - x^{(0)}\|_\infty \leqslant \varepsilon$, 则停算, 输出 x 作为方程组的近似解; 否则, 转步 4.

步 4 若 $k = N$, 则停算, 输出迭代失败信息; 否则置 $x^{(0)} := x$, $k := k + 1$, 转步 2.

根据算法 2.2, 编制 MATLAB 程序如下:

- Gauss-Seidel 迭代法 MATLAB 程序

```
%maseidel.m
function x=maseidel (A,b,x0,ep,N)
%用途: 用Gauss-Seidel迭代法解线性方程组Ax=b
%格式: x=maseidel (A,b,x0,ep,N)  A为系数矩阵, b为右端向量,
%x0为初始向量(默认零向量), ep为精度(默认1e-6), N为最大迭
%代次数(默认500次), x返回近似解向量
n=length(b);
if nargin<5,N=500;end
if nargin<4,ep=1e-6;end
if nargin<3,x0=zeros(n,1);end
x=zeros(n,1); k=0;
while k<N
  for i=1:n
    if i==1
      x(1)=(b(1)-A(1,2:n)*x0(2:n))/A(1,1);
    else if i==n
      x(n)=(b(n)-A(n,1:n-1)*x(1:n-1))/A(n,n);
     else
      x(i)=(b(i)-A(i,1:i-1)*x(1:i-1)-A(i,i+1:n)*x0(i+1:n))/A(i,i);
     end
    end
```

```
    end
    if norm(x-x0,inf)<ep, break; end
    x0=x; k=k+1;
  end
  if k==N,Warning('已达到迭代次数上限');end
  disp(['k=',num2str(k)])
```

例 2.3 用 Gauss-Seidel 迭代法通用程序 maseidel.m 解线性方程组:

$$\begin{pmatrix} 0.76 & -0.01 & -0.14 & -0.16 \\ -0.01 & 0.88 & -0.03 & 0.06 \\ -0.14 & -0.03 & 1.01 & -0.12 \\ -0.16 & 0.06 & -0.12 & 0.72 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 0.68 \\ 1.18 \\ 0.12 \\ 0.74 \end{pmatrix}.$$

取初始点 $x^{(0)} = (0,0,0,0)^{\mathrm{T}}$, 精度要求 $\varepsilon = 10^{-6}$.

解 在 MATLAB 命令窗口执行程序 maseidel.m:

```
>>A=[0.76 -0.01 -0.14 -0.16; -0.01 0.88 -0.03 0.05;
     -0.14 -0.03 1.01 -0.12; -0.16 0.05 -0.12 0.72];
>> b=[0.68 1.18 0.12 0.74]';
>> x=maseidel(A,b)
```

得到计算结果:

```
k = 8
x =
    1.27616296389801
    1.29806390417707
    0.48904229112097
    1.30273326937409
```

2.3.2 收敛性分析

定理 2.8 若线性方程组 (2.1) 的系数矩阵 A 严格对角占优, 即

$$\sum_{j=1,j\neq i}^{n} |a_{ij}| < |a_{ii}|, \quad i=1,2,\cdots,n, \tag{2.17}$$

或

$$\sum_{i=1,i\neq j}^{n} |a_{ij}| < |a_{jj}|, \quad j=1,2,\cdots,n, \tag{2.18}$$

则 Gauss-Seidel 迭代收敛.

证 注意到 Gauss-Seidel 迭代矩阵 $B_S=(D-L)^{-1}U$ 的特征多项式为

$$P(\lambda)=\det[\lambda I-(D-L)^{-1}U]$$

$$=\det\{(D-L)^{-1}[\lambda(D-L)-U]\}$$
$$=\det[(D-L)^{-1}]\cdot\det[\lambda(D-L)-U].$$

首先, $\det[(D-L)^{-1}]\neq 0$. 以下用反证法. 若 Gauss-Seidel 迭代不收敛, 则至少存在一个特征值 λ, 且 $|\lambda|\geqslant 1$. 由于 A 对角占优, 即 (2.17) 或 (2.18) 成立, 故 $\lambda(D-L)-U$ 仍为对角占优, 而对角占优矩阵必非奇异, 故

$$\det[\lambda(D-L)-U]\neq 0,$$

这与 λ 是迭代矩阵 B_S 的特征值相矛盾, 即 $|\lambda|$ 不可大于或等于 1. 因此 $|\lambda|<1$, 即 $\rho(B_S)=\rho((D-L)^{-1}U)<1$, 从而 Gauss-Seidel 迭代收敛. □

定理 2.9 *若线性方程组 (2.1) 的系数矩阵 A 对称正定, 则 Gauss-Seidel 迭代收敛.*

证 记 B_S 的特征值为 λ, 对应的特征向量为 z, 这时

$$(D-L)^{-1}Uz=\lambda z,$$

即

$$Uz=\lambda(D-L)z.$$

上式两边左乘 z 的共轭转置 z^{H} 得

$$z^{\mathrm{H}}Uz=\lambda z^{\mathrm{H}}(D-L)z,$$

即

$$\lambda=\frac{z^{\mathrm{H}}Uz}{z^{\mathrm{H}}Dz-z^{\mathrm{H}}Lz}. \tag{2.19}$$

记 $z^{\mathrm{H}}Dz=d$, $z^{\mathrm{H}}Lz=a+ib$, 因 A 对称, 故 $U=L^{\mathrm{T}}$, $z^{\mathrm{H}}Uz=a-ib$, 代入 (2.19) 得

$$\lambda=\frac{a-ib}{(d-a)-ib}. \tag{2.20}$$

因 A 正定, $z^{\mathrm{H}}Az=z^{\mathrm{H}}(D-L-U)z=d-2a>0$. 由于

$$|\lambda|^2=\frac{a^2+b^2}{(d-a)^2+b^2}=\frac{a^2+b^2}{(a^2+b^2)+d(d-2a)}<1,$$

故迭代收敛. □

注 2.2 *类似于定理 2.7, 定理 2.8 和定理 2.9 也只是 Gauss-Seidel 迭代收敛的充分条件而非必要条件. 请仔细领会下面的例题.*

例 2.4 *判断用 Gauss-Seidel 迭代法解下列方程组的收敛性:*

$$\text{(1)}\ \begin{cases}7x_1-2x_2-3x_3=2.8,\\-x_1+6x_2-2x_3=3.5,\\-x_1-x_2+4x_3=6.2;\end{cases}\qquad \text{(2)}\ \begin{cases}4x_1-2x_2-x_3=1,\\-2x_1+4x_2+3x_3=5,\\-x_1-2x_2+3x_3=0.\end{cases}$$

解 (1) 由于该方程组的系数矩阵严格对角占优, 故由定理 2.8 知, Gauss-Seidel 迭代收敛.

(2) 由于该方程组的系数矩阵既不是严格对角占优矩阵, 也不是对称正定矩阵, 故无法由定理 2.8 或定理 2.9 判断其收敛性. 但其迭代矩阵为

$$B_S = (D-L)^{-1}U = \begin{pmatrix} 4 & 0 & 0 \\ -2 & 4 & 0 \\ -1 & -2 & 3 \end{pmatrix}^{-1} \begin{pmatrix} 0 & 2 & 1 \\ 0 & 0 & -3 \\ 0 & 0 & 0 \end{pmatrix},$$

计算可得

$$B_S = \begin{pmatrix} 0 & 1/2 & 1/4 \\ 0 & 1/4 & -5/8 \\ 0 & 1/3 & -1/3 \end{pmatrix}.$$

从而

$$\|B_S\|_\infty = \max\left\{\frac{3}{4}, \frac{7}{8}, \frac{2}{3}\right\} = \frac{7}{8} < 1,$$

故由定理 2.6 知, Gauss-Seidel 迭代收敛.

例 2.5 *考虑方程组*

$$\begin{cases} x_1 + ax_2 + ax_3 = 2, \\ ax_1 + x_2 + ax_3 = 3, \\ ax_1 + ax_2 + x_3 = 1. \end{cases}$$

(1) *当 a 取何值时, Jacobi 迭代法是收敛的?*

(2) *当 a 取何值时, Gauss-Seidel 迭代法是收敛的?*

解 (1) 容易发现, 只要 $-0.5 < a < 0.5$, 方程组的系数矩阵 A 是严格对角占优的, 故此时 Jacobi 迭代法收敛.

(2) 由 A 的各阶顺序主子式

$$D_1 = 1 > 0, \quad D_2 = 1 - a^2 > 0, \quad D_3 = 1 + 2a^3 - 3a^2 > 0,$$

解得 $-0.5 < a < 1$. 此时矩阵 A 是对称正定的, 故 Gauss-Seidel 迭代法收敛.

2.4 逐次超松弛迭代法

2.4.1 迭代公式及其通用程序

逐次超松弛迭代法可以看作 Gauss-Seidel 迭代法的加速. Gauss-Seidel 迭代格式为

$$x^{(k+1)} = D^{-1}Lx^{(k+1)} + D^{-1}Ux^{(k)} + D^{-1}b,$$

现令

$$\Delta x^{(k)} = x^{(k+1)} - x^{(k)} = D^{-1}Lx^{(k+1)} + D^{-1}Ux^{(k)} + D^{-1}b - x^{(k)},$$

这时

$$x^{(k+1)} = x^{(k)} + \Delta x^{(k)},$$

则 $x^{(k+1)}$ 可以看作由 $x^{(k)}$ 作 $\Delta x^{(k)}$ 修正而得到. 若在修正项中引入一个因子 ω, 即

$$x^{(k+1)} = x^{(k)} + \omega \Delta x^{(k)}, \tag{2.21}$$

即可得到逐次超松弛迭代格式 (SOR). 由 (2.21),

$$x^{(k+1)} = (1-\omega)x^{(k)} + \omega(D^{-1}Lx^{(k+1)} + D^{-1}Ux^{(k)} + D^{-1}b), \tag{2.22}$$

即

$$(I - \omega D^{-1}L)x^{(k+1)} = [(1-\omega)I + \omega D^{-1}U]x^{(k)} + \omega D^{-1}b.$$

故 SOR 迭代的计算格式为

$$x^{(k+1)} = (I - \omega D^{-1}L)^{-1}[(1-\omega)I + \omega D^{-1}U]x^{(k)} + \omega(I - \omega D^{-1}L)^{-1}D^{-1}b. \tag{2.23}$$

SOR 迭代的迭代矩阵为

$$B_\omega = (I - \omega D^{-1}L)^{-1}[(1-\omega)I + \omega D^{-1}U] = (D - \omega L)^{-1}[(1-\omega)D + \omega U].$$

用分量形式表示 (2.22), 即

$$\begin{aligned} x_i^{(k+1)} &= (1-\omega)x_i^{(k)} + \omega\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\right)/a_{ii} \\ &= x_i^{(k)} + \omega\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i}^{n} a_{ij}x_j^{(k)}\right)/a_{ii} \qquad (2.24) \\ &\quad (i = 1, 2, \cdots, n; \quad k = 0, 1, 2, \cdots), \end{aligned}$$

其中, ω 叫松弛因子, 当 $\omega > 1$ 时叫超松弛, $0 < \omega < 1$ 时叫低松弛, $\omega = 1$ 时就是 Gauss-Seidel 迭代法. 下面给出 SOR 迭代的具体算法步骤:

算法 2.3 (SOR 迭代法)

步 1 输入矩阵 A, 右端向量 b, 初始点 $x^{(0)}$, 精度要求 ε, 最大迭代次数 N, 置 $k := 0$;

步 2　计算

$$x_1 = (1-\omega)x_1^{(0)} + \omega\left(b_1 - \sum_{j=2}^{n} a_{1j}x_j^{(0)}\right)/a_{11},$$

$$x_i = (1-\omega)x_i^{(0)} + \omega\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j - \sum_{j=i+1}^{n} a_{ij}x_j^{(0)}\right)/a_{ii}, \quad i=2,\cdots,n-1,$$

$$x_n = (1-\omega)x_n^{(0)} + \omega\left(b_n - \sum_{j=1}^{n-1} a_{nj}x_j\right)/a_{nn};$$

步 3　若 $\|x - x^{(0)}\|_\infty \leqslant \varepsilon$，则停算，输出 x 作为方程组的近似解；否则，转步 4.

步 4　若 $k = N$，则停算，输出迭代失败信息；否则置 $x^{(0)} := x$，$k := k+1$，转步 2.

根据算法 2.3，编制 MATLAB 程序如下：

- SOR 迭代法 MATLAB 程序

```
%masor.m
function x=masor(A,b,omega,x0,ep,N)
%用途：用SOR迭代法解线性方程组Ax=b
%格式：x=masor(A,b,omega,x0,ep,N)  A为系数矩阵，b为右端向量，
%omega为松弛因子(默认1.5)，x0为初始向量(默认零向量)，ep为精
%度(默认1e-6)，N为最大迭代次数(默认500次)，x返回近似解向量
n=length(b);
if nargin<6,N=500;end
if nargin<5,ep=1e-6;end
if nargin<4,x0=zeros(n,1);end
if nargin<3,omega=1.5;end
x=zeros(n,1); k=0;
 while k<N
  for i=1:n
   if i==1
      x1(1)=(b(1)-A(1,2:n)*x0(2:n))/A(1,1);
   else if i==n
      x1(n)=(b(n)-A(n,1:n-1)*x(1:n-1))/A(n,n);
     else
      x1(i)=(b(i)-A(i,1:i-1)*x(1:i-1)-A(i,i+1:n)*x0(i+1:n))/A(i,i);
     end
```

```
        end
        x(i)=(1-omega)*x0(i)+omega*x1(i);
      end
      if norm(x0-x,inf)<ep, break; end
      k=k+1;   x0=x;
    end
    if k==N,Warning('已达到迭代次数上限');end
    disp(['k=',num2str(k)])
```

例 2.6 用 SOR 迭代法程序 masor.m 解线性方程组:

$$\begin{pmatrix} 0.76 & -0.01 & -0.14 & -0.16 \\ -0.01 & 0.88 & -0.03 & 0.06 \\ -0.14 & -0.03 & 1.01 & -0.12 \\ -0.16 & 0.06 & -0.12 & 0.72 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 0.68 \\ 1.18 \\ 0.12 \\ 0.74 \end{pmatrix}.$$

取初始点 $x^{(0)}=(0,0,0,0)^{\mathrm{T}}$, 松弛因子 $\omega=1.05$, 精度要求 $\varepsilon=10^{-6}$.

解 在 MATLAB 命令窗口执行程序 masor.m:

```
>>A=[0.76 -0.01 -0.14 -0.16; -0.01 0.88 -0.03 0.05;
    -0.14 -0.03 1.01 -0.12; -0.16 0.05 -0.12 0.72];
>> b=[0.68 1.18 0.12 0.74]';
>> x=masor(A,b,1.05)
```

得计算结果如下:

```
k = 6
x =
   1.27616302863910
   1.29806392444062
   0.48904230122688
   1.30273328637534
```

2.4.2 收敛性分析

下面给出 SOR 迭代法的收敛性结果.

定理 2.10 SOR 迭代 (2.23) 收敛的必要条件是 $0<\omega<2$.

证 SOR 迭代矩阵为 B_ω, 若 SOR 迭代收敛, 则 $\rho(B_\omega)<1$. 从而

$$|\det(B_\omega)|=|\lambda_1\lambda_2\cdots\lambda_n|<1,$$

这里, $\lambda_1,\lambda_2,\cdots,\lambda_n$ 为 B_ω 的特征值. 又

$$|\det(B_\omega)|=|\det(I-\omega D^{-1}L)^{-1}|\cdot|\det[(1-\omega)I+\omega D^{-1}U]|<1,$$

这里, $I-\omega D^{-1}L$ 是单位下三角矩阵, 而 $(1-\omega)I+\omega D^{-1}U$ 是上三角矩阵, 其对角元均为 $1-\omega$, 故

$$|\det(B_\omega)|=|(1-\omega)^n|,$$

得

$$|1-\omega|<1,$$

即 $0<\omega<2$. □

定理 2.11 若线性方程组 (2.1) 的系数矩阵 A 对称正定, 则当 $0<\omega<2$ 时, SOR 迭代 (2.23) 收敛.

证 记 B_ω 的特征值为 λ, 对应的特征向量为 z, 这时

$$(I-\omega D^{-1}L)^{-1}[(1-\omega)I+\omega D^{-1}U]z=\lambda z,$$

即

$$[(1-\omega)I+\omega D^{-1}U]z=\lambda(I-\omega D^{-1}L)z.$$

上式两边左乘 z 的共轭转置 z^{H} 得

$$(1-\omega)z^{\mathrm{H}}Dz+\omega z^{\mathrm{H}}Uz=\lambda(z^{\mathrm{H}}Dz-\omega z^{\mathrm{H}}Lz),$$

即

$$\lambda=\frac{(1-\omega)z^{\mathrm{H}}Dz+\omega z^{\mathrm{H}}Uz}{z^{\mathrm{H}}Dz-\omega z^{\mathrm{H}}Lz}. \tag{2.25}$$

记 $z^{\mathrm{H}}Dz=d$, $z^{\mathrm{H}}Lz=a+ib$, 因 A 对称, 故 $U=L^{\mathrm{T}}$, $z^{\mathrm{H}}Uz=a-ib$, 代入 (2.25) 得

$$\lambda=\frac{(1-\omega)d+\omega(a-ib)}{d-\omega(a+ib)}=\frac{[(1-\omega)d+\omega a]-i\omega b}{(d-\omega a)-i\omega b}. \tag{2.26}$$

因 A 正定, 故 $z^{\mathrm{H}}Az=z^{\mathrm{H}}(D-L-U)z=d-2a>0$. 注意到 λ 的分子、分母虚部相等, 而当 $0<\omega<2$ 时, 有

$$(d-\omega a)^2-[(1-\omega)d+\omega a]^2=(2-\omega)\omega d(d-2a)>0.$$

由此可得 $|\lambda|<1$, 故迭代收敛. □

推论 2.1 A 对称正定时, Jacobi 迭代收敛的充要条件是 $2D-A$ 也对称正定. (证明略.)

例 2.7 已知矩阵 A 如下, 判断求解 $Ax=b$ 的 Jacobi 迭代法、Gauss-Seidel 及 SOR 迭代法是否收敛:

$$(1)\ A=\begin{pmatrix}1&-1&2\\-1&3&0\\2&0&7\end{pmatrix},\qquad (2)\ A=\begin{pmatrix}2&1&1\\1&2&1\\1&1&2\end{pmatrix}.$$

解 (1) 显然 A 是对称的, 顺序主子式

$$D_1=1>0,\quad D_2=\begin{vmatrix}1 & -1\\ -1 & 3\end{vmatrix}=2>0,\quad D_3=\begin{vmatrix}1 & -1 & 2\\ -1 & 3 & 0\\ 2 & 0 & 7\end{vmatrix}=2>0,$$

故 A 对称正定. 从而由定理 2.11, 当 $0<\omega<2$ 时, SOR 迭代法是收敛的. 由于 Gauss-Seidel 迭代是 SOR 迭代 $\omega=1$ 时的特例, 故 Gauss-Seidel 迭代也是收敛的. 又

$$2D-A=\begin{pmatrix}1 & 1 & -2\\ 1 & 3 & 0\\ -2 & 0 & 7\end{pmatrix}$$

也显然是对称的, 且其顺序主子式的值与 A 的相同, 故 $2D-A$ 也是对称正定的, 从而由推论 2.1 知, Jacobi 迭代也是收敛的.

(2) 容易验证 A 是对称正定的, 故 Gauss-Seidel 迭代和 SOR 迭代当 $0<\omega<2$ 时都是收敛的. 当 $\det(2D-A)=0$, 故 $2D-A$ 不正定, 故由推论 2.1 知, Jacobi 迭代是发散的.

习 题 2

(Ⅰ) 理论分析题

2.1 证明对 n 阶非奇异矩阵 A 和 n 阶奇异矩阵 B, 有 $\|A^{-1}\|\geqslant\dfrac{1}{\|A-B\|}$.

2.2 已知向量 $x=(2,\ -3,\ 4)^{\mathrm T}$, 矩阵 $A=\begin{pmatrix}1 & 0 & 0\\ 0 & 2 & 4\\ 0 & -2 & 4\end{pmatrix}$, 求向量 x 和矩阵 A 的三种常用范数.

2.3 设有线性方程组

$$\begin{cases}x_1+0.4x_2+0.4x_3=1,\\ 0.4x_1+x_2+0.8x_3=2,\\ 0.4x_1+0.8x_2+x_3=3,\end{cases}$$

试考察此方程组的 Jacobi 迭代法的收敛性.

2.4 设有线性方程组

$$\begin{cases}x_1+2x_2-2x_3=1,\\ x_1+x_2+x_3=1,\\ 2x_1+2x_2+x_3=1,\end{cases}$$

试考察此方程组的 Gauss-Seidel 迭代法的收敛性.

2.5 设有线性方程组 $Ax = b$, 其中系数矩阵

$$A = \begin{pmatrix} 4 & -1 & & \\ -1 & 4 & -1 & \\ & -1 & 4 & -1 \\ & & -1 & 4 \end{pmatrix}.$$

证明用 Jacobi 迭代法, Gauss-Seidel 迭代法, SOR 法求解上述方程组都是收敛的.

2.6 设有线性方程组 $Ax = b$, 其中系数矩阵

$$A = \begin{pmatrix} 1 & 2 & -2 \\ 1 & 1 & 1 \\ 2 & 2 & 1 \end{pmatrix}.$$

证明 Jacobi 迭代法收敛, 而 Gauss-Seidel 迭代法发散.

2.7 设有线性方程组 $Ax = b$, 其中系数矩阵

$$A = \begin{pmatrix} 2 & -1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & -2 \end{pmatrix}.$$

证明 Gauss-Seidel 迭代法收敛, 而 Jacobi 迭代法发散.

2.8 设系数矩阵

$$A = \begin{pmatrix} a & 1 & 3 \\ 1 & a & 2 \\ -3 & 2 & a \end{pmatrix}.$$

(1) 当 a 取何值时, Jacobi 迭代法收敛?

(2) 当 a 取何值时, Gauss-Seidel 迭代法收敛?

2.9 设有方程组

$$\begin{cases} 9x_1 + 2x_2 + 2x_3 = 7, \\ 2x_2 + 5x_2 + 4x_3 = -2, \\ 2x_1 + 4x_2 + 5x_3 = -2, \end{cases}$$

写出 Jacobi 迭代、Gauss-Seidel 迭代和 SOR 迭代 ($\omega = 1.32$) 的迭代公式, 并讨论三种迭代格式的收敛性.

2.10 设 $A \in \mathbf{R}^{n\times n}$ 对称正定, 其最小特征值和最大特征值分别为 λ_1, λ_n, 证明迭代法

$$x^{(k+1)} = x^{(k)} + \theta(b - Ax^{(k)})$$

收敛的充分必要条件是 $0 < \theta < 2/\lambda_n$.

2.11 用 Gauss-Seidel 迭代法求解线性方程组

$$\begin{cases} x_1 + ax_2 = -2, \\ 2ax_1 + x_2 = 1, \end{cases}$$

问当 a 取何值时, 迭代格式是收敛的.

2.12 分别写出解线性方程组

$$\begin{cases} x_1 - 5x_2 + x_3 = 14, \\ x_1 + x_2 - 4x_3 = -13, \\ -8x_1 + x_2 + x_3 = -7 \end{cases}$$

收敛的 Jacobi 迭代格式与 Gauss-Seidel 迭代格式.

2.13 设 $x = Jx + f$, 其中

$$J = \begin{pmatrix} 0.9 & 0 \\ 0.3 & 0.8 \end{pmatrix}, \quad f = \begin{pmatrix} 1 \\ 2 \end{pmatrix}.$$

证明虽然 $\|J\| > 1$, 但迭代法 $x^{(k+1)} = Jx^{(k)} + f$ 收敛.

2.14 证明给定线性方程组雅可比迭代发散, 而高斯–赛德尔迭代收敛:

$$\begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

2.15 证明给定线性方程组雅可比迭代收敛, 而高斯–赛德尔迭代发散:

$$\begin{pmatrix} 1 & 0 & 1 \\ -1 & 1 & 0 \\ 1 & 2 & -3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}.$$

2.16 已知线性方程组

$$\begin{cases} 11x_1 - 5x_2 - 33x_3 = 1, \\ -22x_1 + 11x_2 + x_3 = 0, \\ x_1 - 4x_2 + 2x_3 = 1. \end{cases}$$

用两种不同的方法判别其迭代的收敛性.

(Ⅱ) 上机实验题

2.1 利用算法 2.1 (Jacobi 迭代法), 编制 MATLAB 程序, 求线性方程组

(1) $$\begin{pmatrix} 14 & 4 & 4 & 4 \\ 4 & 14 & 4 & 4 \\ 4 & 4 & 14 & 4 \\ 4 & 4 & 4 & 14 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} -4 \\ 16 \\ 36 \\ 56 \end{pmatrix};$$

(2) $$\begin{pmatrix} 10.9 & 1.2 & 2.1 & 0.9 \\ 1.2 & 11.2 & 1.5 & 2.5 \\ 2.1 & 1.5 & 9.8 & 1.3 \\ 0.9 & 2.5 & 1.3 & 12.3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} -7.0 \\ 5.3 \\ 10.3 \\ 24.6 \end{pmatrix}$$

的近似解, 取初值 $x=(0,0,0,0)^{\mathrm{T}}$.

2.2 利用算法 2.2 (Gauss-Seidel 迭代法), 编制 MATLAB 程序, 求线性方程组

(1) $$\begin{pmatrix} 6 & -2 & -1 & -1 \\ -2 & 12 & -1 & -1 \\ -1 & -1 & 6 & -2 \\ -1 & -1 & -1 & 12 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}=\begin{pmatrix} -16 \\ 6 \\ 8 \\ 54 \end{pmatrix};$$

(2) $$\begin{pmatrix} 0.78 & -0.02 & -0.12 & -0.14 \\ -0.02 & 0.86 & -0.04 & -0.06 \\ -0.12 & -0.04 & 0.72 & -0.08 \\ -0.14 & -0.06 & -0.08 & 0.74 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}=\begin{pmatrix} 0.76 \\ 0.08 \\ 1.12 \\ 0.68 \end{pmatrix}$$

的近似解, 取初值 $x=(0,0,0,0)^{\mathrm{T}}$.

2.3 利用算法 2.3 (SOR 法), 编制 MATLAB 程序, 求线性方程组

(1) $$\begin{pmatrix} -4 & 1 & 1 & 1 \\ 1 & -4 & 1 & 1 \\ 1 & 1 & -4 & 1 \\ 1 & 1 & 1 & -4 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}=\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix};$$

(2) $$\begin{pmatrix} 1 & 0 & -0.25 & -0.25 \\ 0 & 1 & -0.25 & -0.25 \\ -0.25 & -0.25 & 1 & 0 \\ -0.25 & -0.25 & 0 & 1 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}=\begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{pmatrix}$$

的近似解, 取初值 $x=(0,0,0,0)^{\mathrm{T}}$, 松弛因子分别为 $\omega=1.3,\ \omega=1.1$.

第 3 章　解线性方程组的直接法

本章研究 n 阶线性方程组的直接解法

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2, \\ \cdots\cdots\cdots\cdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n. \end{cases} \tag{3.1}$$

若用矩阵和向量的记号来表示, (3.1) 可写成

$$Ax = b, \tag{3.2}$$

其中, n 阶矩阵 $A = (a_{ij})_{n\times n}$ 称为方程组的系数矩阵, n 维向量 $b = (b_1, b_2, \cdots, b_n)^{\mathrm{T}}$ 称为右端项, $x = (x_1, x_2, \cdots, x_n)^{\mathrm{T}}$ 为所求的解. 所谓直接法, 是指经过有限步运算后能求得方程组精确解的方法. 若 A 非奇异, 方程组 (3.1) 有唯一解. 下面介绍几种比较实用的直接法.

3.1　顺序 Gauss 消去法及其程序实现

Gauss 消去法的基本思想是: 首先使用初等行变换将方程组转化为一个同解的上三角形方程组 (称为消元), 再通过回代法求解该三角形方程组 (称为回代). 按行原先的位置进行消元的 Gauss 消去法成为顺序 Gauss 消去法.

例 3.1　用顺序 Gauss 消去法解线性方程组

$$\begin{cases} x_1 + x_2 + x_3 + x_4 = 10, \\ -x_1 + 2x_2 - 3x_3 + x_4 = -2, \\ 3x_1 - 3x_2 + 6x_3 - 2x_4 = 7, \\ -4x_1 + 5x_2 + 2x_3 - 3x_4 = 0. \end{cases}$$

解　1. 消元过程:

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 10 \\ -1 & 2 & -3 & 1 & -2 \\ 3 & -3 & 6 & -2 & 7 \\ -4 & 5 & 2 & -3 & 0 \end{pmatrix} \xrightarrow[\substack{r_3-3r_1 \\ r_4+4r_1}]{r_2+r_1} \begin{pmatrix} 1 & 1 & 1 & 1 & 10 \\ 0 & 3 & -2 & -2 & 8 \\ 0 & -6 & 3 & -5 & -23 \\ 0 & 9 & 6 & 1 & 40 \end{pmatrix} \xrightarrow[r_4-3r_2]{r_3+2r_2}$$

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 10 \\ 0 & 3 & -2 & -2 & 8 \\ 0 & 0 & -1 & -1 & -7 \\ 0 & 0 & 12 & -5 & 16 \end{pmatrix} \xrightarrow{r_4+12r_3} \begin{pmatrix} 1 & 1 & 1 & 1 & 10 \\ 0 & 3 & -2 & -2 & 8 \\ 0 & 0 & -1 & -1 & -7 \\ 0 & 0 & 0 & -17 & -68 \end{pmatrix}.$$

2. 回代过程:

$$\begin{cases} x_1 + x_2 + x_3 + x_4 = 10, \\ \quad 3x_2 - 2x_3 + 2x_4 = 8, \\ \quad\quad -x_3 - x_4 = -7, \\ \quad\quad\quad -17x_4 = -68 \end{cases} \Rightarrow \begin{cases} x_4 = 4, \\ x_3 = 7 - x_4 = 3, \\ x_2 = (8 + 2x_3 - 2x_4)/3 = 2, \\ x_1 = 10 - x_1 - x_2 - x_3 = 1. \end{cases}$$

对于一般线性方程组, 使用顺序 Gauss 消去法求解

$$\begin{cases} a_{11}^{(1)}x_1 + a_{12}^{(1)}x_2 + \cdots + a_{1n}^{(1)}x_n = b_1^{(1)}, \\ a_{21}^{(1)}x_1 + a_{22}^{(1)}x_2 + \cdots + a_{2n}^{(1)}x_n = b_2^{(1)}, \\ \cdots\cdots\cdots\cdots \\ a_{n1}^{(1)}x_1 + a_{n2}^{(1)}x_2 + \cdots + a_{1n}^{(1)}x_n = b_n^{(1)}. \end{cases} \tag{3.3}$$

1. 消元过程:

$$\begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & a_{23}^{(1)} & \cdots & a_{2n}^{(1)} & b_2^{(1)} \\ a_{31}^{(1)} & a_{32}^{(1)} & a_{33}^{(1)} & \cdots & a_{3n}^{(1)} & b_3^{(1)} \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ a_{n1}^{(1)} & a_{n2}^{(1)} & a_{n3}^{(1)} & \cdots & a_{nn}^{(1)} & b_n^{(1)} \end{pmatrix} \rightarrow \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} & b_2^{(2)} \\ 0 & a_{32}^{(2)} & a_{33}^{(2)} & \cdots & a_{3n}^{(2)} & b_3^{(2)} \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & a_{n2}^{(2)} & a_{n3}^{(2)} & \cdots & a_{nn}^{(2)} & b_n^{(2)} \end{pmatrix}$$

$$\rightarrow \cdots \rightarrow \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} & b_2^{(2)} \\ 0 & 0 & a_{33}^{(3)} & \cdots & a_{3n}^{(3)} & b_3^{(3)} \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a_{nn}^{(n)} & b_n^{(n)} \end{pmatrix},$$

其中

$$a_{ij}^{(2)} = a_{ij}^{(1)} - m_{i1}a_{1j}^{(1)}, \quad b_i^{(2)} = b_i^{(1)} - m_{i1}b_1^{(1)}, \quad m_{i1} = \frac{a_{i1}^{(1)}}{a_{11}^{(1)}}, \quad i, j = 2, \cdots, n.$$

一般地,

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik}a_{kj}^{(k)}, \quad b_i^{(k+1)} = b_i^{(k)} - m_{ik}b_k^{(k)}, \quad m_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}},$$
$$i, j = k+1, \cdots, n; \ k = 1, \cdots, n-1. \tag{3.4}$$

2. 回代过程:

$$\begin{cases} a_{11}^{(1)}x_1 + a_{12}^{(1)}x_2 + \cdots + a_{1n}^{(1)}x_n = b_1^{(1)}, \\ \qquad a_{22}^{(2)}x_2 + \cdots + a_{2n}^{(2)}x_n = b_2^{(2)}, \\ \qquad \cdots\cdots\cdots\cdots \\ \qquad\qquad a_{nn}^{(n)}x_n = b_n^{(n)}, \end{cases}$$

$$\Longrightarrow \begin{cases} x_n = b_n^{(n)}/a_{nn}^{(n)}, \\ x_k = \left(b_k^{(k)} - \sum_{j=k+1}^{n} a_{kj}^{(k)}x_j\right)/a_{kk}^{(k)}, \quad k = n-1, \cdots, 2, 1. \end{cases} \tag{3.5}$$

在此基础上, 我们得到顺序 Gauss 消去法的算法步骤:

算法 3.1 (顺序 Gauss 消去法)

步 1 输入系数矩阵 A, 右端项 b, 置 $k := 1$;

步 2 消元: 对 $k = 1, \cdots, n-1$, 计算

$$m_{ik} = a_{ik}^{(k)}/a_{kk}^{(k)}, \quad a_{ik}^{(k+1)} = 0,$$
$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik}a_{kj}^{(k)}, \quad b_i^{(k+1)} = b_i^{(k)} - m_{ik}b_k^{(k)}.$$
$$(i = k+1, \cdots, n; \quad j = k+1, \cdots, n.)$$

步 3 回代:

$$x_n = b_n^{(n)}/a_{nn}^{(n)},$$

对 $k = n-1, \cdots, 1$, 计算

$$x_k = \left(b_k^{(k)} - \sum_{j=k+1}^{n} a_{kj}^{(k)}x_j\right)/a_{kk}^{(k)}.$$

现在我们来统计顺序 Gauss 消去法的计算量. 由于加减法的计算量可忽略不计, 我们只统计乘除法次数.

消元过程: 第 k $(k = 1, \cdots, n-1)$ 步消元有

$$(n-k)(n-k+1) + (n-k) = (n-k)(n-k+2)$$

次乘除法, 共

$$N_1 = \sum_{k=1}^{n-1}(n-k)(n-k+2) = \sum_{i=1}^{n-1}(i^2+2i)$$
$$= \frac{n(n-1)(2n-1)}{6} + n(n-1) = \frac{n(n-1)(2n+5)}{6}$$

次乘除法.

回代过程: 计算 x_k $(k = n, \cdots, 2, 1)$ 时, 有 $n-k+1$ 次乘除法, 共

$$N_2 = \sum_{k=1}^{n}(n-k+1) = \sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

次乘除法.

消元和回代过程共计

$$N_1 + N_2 = \frac{n(n-1)(2n+5)}{6} + \frac{n(n+1)}{2} = \frac{n^3}{3} + n^2 - \frac{n}{3}$$

次乘除法.

可见消元过程的计算量为 $O(n^3)$, 而回代过程的计算量为 $O(n^2)$, 因此顺序 Gauss 消去法的计算量主要在消元过程部分.

可以证明, 如果 $A = (a_{ij})_{n\times n}$ 的顺序主子式

$$D_1 = a_{11}, \quad D_2 = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}, \quad \cdots, \quad D_n = \begin{vmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{vmatrix}$$

均不为 0, 则算法 3.1 是可行的.

根据算法 3.1, 编制 MATLAB 程序如下:

- *顺序 Gauss 消去法 MATLAB 程序*

```
%magauss.m
function x=magauss(A,b,flag)
%用途：顺序Gauss消去法解线性方程组Ax=b
%格式：x=magauss(A,b,flag)，A为系数矩阵，b为右端项，若flag=0,
%则不显示中间过程，否则显示中间过程，默认为0，x为解向量
if nargin<3,flag=0;end
n=length(b);
%消元
for k=1:(n-1)
    m=A(k+1:n,k)/A(k,k);
    A(k+1:n,k+1:n)=A(k+1:n,k+1:n)-m*A(k,k+1:n);
    b(k+1:n)=b(k+1:n)-m*b(k);
    A(k+1:n,k)=zeros(n-k,1);
```

```
    if flag~=0, Ab=[A,b], end
end
%回代
x=zeros(n,1);
x(n)=b(n)/A(n,n);
for k=n-1:-1:1
    x(k)=(b(k)-A(k,k+1:n)*x(k+1:n))/A(k,k);
end
```

例 3.2 *利用通用程序 magauss.m 计算下列方程组的解:*

$$\begin{cases} x_1+x_2+x_3+x_4=10, \\ -x_1+2x_2-3x_3+x_4=-2, \\ 3x_1-3x_2+6x_3-2x_4=7, \\ -4x_1+5x_2+2x_3-3x_4=0. \end{cases}$$

解 在 MATLAB 命令窗口执行

```
>> A=[1 1 1 1;-1 2 -3 1;3 -3 6 -2;-4 5 2 -3];
>> b=[10 -2 7 0]';
>> x=magauss(A,b); x'
```

得计算结果:

```
x =
   1  2  3   4
```

例 3.3 *证明: 顺序 Gauss 消去法可行的充分必要条件是系数矩阵 A 的所有顺序主子式 $D_i\neq 0, i=1,2,\cdots,n$.*

证 必要性. 若顺序 Gauss 消去法是可行的, 即 $a_{ii}^{(i)}\neq 0$, 则可进行消去法的 $k-1$ 步 $(k\leqslant n)$. 由于 $A^{(k)}$ 是由 A 逐行实行初等变换 (某数乘以某一行加到另一行) 得到的, 这些运算不改变相应顺序主子式的值, 故有

$$D_k=\begin{vmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1k}^{(1)} \\ a_{21}^{(2)} & a_{22}^{(2)} & \cdots & a_{2k}^{(2)} \\ & & \ddots & \vdots \\ & & & a_{kk}^{(k)} \end{vmatrix}=a_{11}^{(1)}a_{22}^{(2)}\cdots a_{kk}^{(k)}\neq 0,\quad k=1,2,\cdots,n.$$

充分性. 用归纳法证明. 当 $k=1$ 时显然成立. 设命题对 $k-1$ 成立. 现设 $D_1\neq 0,\cdots,D_{k-1}\neq 0, D_k\neq 0$. 由归纳法假设有 $a_{11}^{(1)}\neq 0,\cdots,a_{k-1,k-1}^{(k-1)}\neq 0$. 因此, 消去法可以进行第 $k-1$ 步, A 约化为

$$A^{(k)}=\begin{pmatrix} A_{11}^{(k-1)} & A_{12}^{(k-1)} \\ & A_{22}^{(k)} \end{pmatrix},$$

其中 $A_{11}^{(k-1)}$ 是对角元为 $a_{11}^{(1)}, \cdots, a_{k-1,k-1}^{(k-1)}$ 的上三角矩阵, 因 $A^{(k)}$ 是通过行初等变换由 A 逐步得到的, 故 A 的 k 阶顺序主子式与 $A^{(k)}$ 的 k 阶顺序主子式相等, 即

$$D_k = \det \begin{pmatrix} A_{11}^{(k-1)} & A_{12}^{(k-1)} \\ & a_{kk}^{(k)} \end{pmatrix} = a_{11}^{(1)} \cdots a_{k-1,k-1}^{(k-1)} a_{kk}^{(k)}.$$

故由 $D_k \neq 0$ 及归纳法假设可推出 $a_{kk}^{(k)} \neq 0$. □

3.2 列主元 Gauss 消去法及程序实现

顺序 Gauss 消去法的计算过程是不可靠的, 一旦出现 $a_{kk}^{(k)} = 0$, 计算就无法进行下去. 即使对所有 $k = 1, 2, \cdots, n$, $a_{kk}^{(k)} \neq 0$, 也不能保证计算过程是数值稳定的.

例 3.4 设有线性方程组

$$\begin{cases} 0.0001x_1 + 1.0x_2 = 1.0, \\ 1.0x_1 + 1.0x_2 = 2.0, \end{cases}$$

其精确解为

$$x_1 = \frac{10000}{9999} \approx 1.00010, \quad x_2 = \frac{9998}{9999} \approx 0.99990.$$

现在假定用尾数为 4 位十进制字长的浮点数来求解.

解 消元过程: 根据 4 位浮点数运算规则 $1.0 - 10000.0 = (0.00001 - 0.1)10^5 = (0.0000 - 0.1)10^5 = -10000.0$ (舍入), 同理, $2.0 - 10000.0 = -10000.0$,

$$\begin{pmatrix} 0.0001 & 1.0 & 1.0 \\ 1.0 & 1.0 & 2.0 \end{pmatrix} \xrightarrow{r_2 - 10^4 r_1} \begin{pmatrix} 0.0001 & 1.0 & 1.0 \\ 0 & 1.0 - 10000.0 & 2.0 - 10000.0 \end{pmatrix}$$

$$\xrightarrow{\text{舍入}} \begin{pmatrix} 0.0001 & 1.0 & 1.0 \\ 0 & -10000.0 & -10000.0 \end{pmatrix}.$$

回代过程:

$$\begin{cases} 0.0001x_1 + 1.0x_2 = 1.0, \\ \qquad -10000.0x_2 = -10000.0. \end{cases} \Longrightarrow \begin{cases} x_2 = 1.0, \\ x_1 = 0.0. \end{cases}$$

代入原方程组验算, 发现结果严重失真.

分析结果失真的原因发现, 由于第一列的主元素 0.0001 绝对值过于小, 当它在消元过程中作分母时把中间过程数据放大 10000 倍, 使中间结果“吃”掉了原始数据, 从而造成数值不稳定.

针对以上问题, 考虑选用绝对值大的数作为主元素.

消元过程：

$$\begin{pmatrix} 0.0001 & 1.0 & 1.0 \\ 1.0 & 1.0 & 2.0 \end{pmatrix} \xrightarrow{r_1 \leftrightarrow r_2} \begin{pmatrix} 1.0 & 1.0 & 2.0 \\ 0.0001 & 1.0 & 1.0 \end{pmatrix}$$

$$\xrightarrow{r_2 - 0.0001r_1} \begin{pmatrix} 1.0 & 1.0 & 2.0 \\ 0 & 1.0-0.0001 & 1.0-0.0002 \end{pmatrix}$$

$$\xrightarrow{\text{舍入}} \begin{pmatrix} 1.0 & 1.0 & 2.0 \\ 0 & 1.0 & 1.0 \end{pmatrix}.$$

这里, 舍入过程 $1.0-0.0001=(0.1-0.00001)10^1$ (舍入), 同理 $1.0-0.0002=1.0$.

回代过程：

$$\begin{cases} 1.0x_1 + 1.0x_2 = 2.0, \\ 1.0x_2 = 1.0. \end{cases} \Longrightarrow \begin{cases} x_2 = 1.0, \\ x_1 = 1.0. \end{cases}$$

代入原方程组验算, 发现结果基本合理.

上述例子说明了选主元素的重要性. 下面阐述列主元 Gauss 消去法的基本思想. 记 $A^{(1)}=A$, 在 Gauss 消元过程的第 1 步, 取的第 1 列中绝对值最大的元素 $a_{r_1 1}^{(1)}$, 即

$$a_{r_1 1}^{(1)} = \max_{1 \leqslant i \leqslant n} |a_{i1}^{(1)}|$$

作为主元素. 若 $r_1 > 1$, 交换第 r_1 行和第 1 行.

一般地, 在 Gauss 消元过程的第 k 步, 取

$$a_{r_k k}^{(k)} = \max_{k \leqslant i \leqslant n} |a_{ik}^{(k)}| \tag{3.6}$$

作为主元素. 若 $r_k > k$, 交换第 r_k 行和第 k 行.

列主元 Gauss 消去法算法步骤如下：

算法 3.2（列主元 Gauss 消去法）

步 1 输入系数矩阵 A, 右端项 b, 置 $k:=1$;

步 2 对 $k=1,\cdots,n-1$ 进行如下操作:

(1) 选列主元, 确定 r_k, 使

$$a_{r_k k}^{(k)} = \max_{k \leqslant i \leqslant n} |a_{ik}^{(k)}|,$$

若 $a_{r_k k}^{(k)} = 0$, 则停止计算, 否则, 进行下一步;

(2) 若 $r_k > k$, 交换 $(A^{(k)}, b^{(k)})$ 的第 k, r_k 两行;

(3) 消元: 对 $i,j=k+1,\cdots,n$, 计算

$$m_{ik} = a_{ik}^{(k)}/a_{kk}^{(k)}, \quad a_{ik}^{(k+1)} = 0,$$

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik}a_{kj}^{(k)}, \quad b_i^{(k+1)} = b_i^{(k)} - m_{ik}b_k^{(k)};$$

步 3 回代

$$x_n = b_n^{(n)}/a_{nn}^{(n)},$$

对 $k = n-1, \cdots, 1$, 计算

$$x_k = \left(b_k^{(k)} - \sum_{j=k+1}^{n} a_{kj}^{(k)} x_j\right) / a_{kk}^{(k)}.$$

根据算法 3.2, 给出 MATLAB 程序如下:

- 列主元 Gauss 消去法 MATLAB 程序

```
%magauss2.m
function x=magauss2(A,b,flag)
%用途: 列主元Gauss消去法解线性方程组Ax=b
%格式: x=magauss(A,b,flag), A为系数矩阵, b为右端项, 若flag=0,
%则不显示中间过程, 否则显示中间过程, 默认为0, x为解向量
if nargin<3,flag=0;end
n=length(b);
for k=1:(n-1)  % 选主元
   [ap,p]=max(abs(A(k:n,k)));
   p=p+k-1;
   if p>k
      t=A(k,:); A(k,:)=A(p,:); A(p,:)=t;
      t=b(k); b(k)=b(p); b(p)=t;
   end
   % 消元
   m=A(k+1:n,k)/A(k,k);
   A(k+1:n,k+1:n)=A(k+1:n,k+1:n)-m*A(k,k+1:n);
   b(k+1:n)=b(k+1:n)-m*b(k);  A(k+1:n,k)=zeros(n-k,1);
   if flag~=0, Ab=[A,b], end
end
% 回代
x=zeros(n,1);
x(n)=b(n)/A(n,n);
for k=n-1:-1:1
   x(k)=(b(k)-A(k,k+1:n)*x(k+1:n))/A(k,k);
end
```

例 3.5 利用程序 magauss2.m 计算下列线性方程组的解

$$\begin{pmatrix} 2 & -1 & 4 & -3 & 1 \\ -1 & 1 & 2 & 1 & 3 \\ 4 & 2 & 3 & 3 & -1 \\ -3 & 1 & 3 & 2 & 4 \\ 1 & 3 & -1 & 4 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 11 \\ 14 \\ 4 \\ 16 \\ 18 \end{pmatrix}.$$

解 在 MATLAB 命令窗口执行

```
>> A=[2 -1 4 -3 1;-1 1 2 1 3;4 2 3 3 -1;-3 1 3 2 4;1 3 -1 4 4];
>> b=[11 14 4 16 18]';
>> x=magauss2(A,b); x'
```

得计算结果:

```
x =
   1.0000   2.0000   1.0000  -1.0000   4.0000
```

3.3 解三对角方程组的追赶法

在科学与工程计算中, 经常遇到求解三对角方程组的问题:

$$\begin{pmatrix} b_1 & c_1 & & & \\ a_2 & b_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & a_n & b_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_{n-1} \\ d_n \end{pmatrix}. \tag{3.7}$$

将 Gauss 消去法应用于三对角方程组得到所谓“追赶法”. 具体操作过程为:

追:

$$\begin{pmatrix} b_1 & c_1 & & & & d_1 \\ a_2 & b_2 & c_2 & & & d_2 \\ & \ddots & \ddots & \ddots & & \vdots \\ & & a_{n-1} & b_{n-1} & c_{n-1} & d_{n-1} \\ & & & a_n & b_n & d_n \end{pmatrix} \to \begin{pmatrix} \bar{b}_1 & c_1 & & & & \bar{d}_1 \\ & \bar{b}_2 & c_2 & & & \bar{d}_2 \\ & & \ddots & \ddots & & \vdots \\ & & & \bar{b}_{n-1} & c_{n-1} & \bar{d}_{n-1} \\ & & & & \bar{b}_n & \bar{d}_n \end{pmatrix},$$

其中

$$\begin{cases} \bar{b}_1 = b_1, \quad \bar{d}_1 = d_1, \\ \bar{b}_k = b_k - \dfrac{a_k}{b_{k-1}} c_{k-1}, \quad k = 2, \cdots, n. \\ \bar{d}_k = d_k - \dfrac{a_k}{b_{k-1}} \bar{d}_{k-1}. \end{cases} \tag{3.8}$$

赶：

$$x_n = \frac{\bar{d}_n}{\bar{b}_n}, \quad x_k = \frac{\bar{d}_k - c_k x_{k+1}}{\bar{b}_k}, \quad k = n-1, \cdots, 2, 1. \tag{3.9}$$

追赶法不需要对零元素计算, 只有 $6n-5$ 次乘除法计算量, 且当系数矩阵对角占优时数值稳定, 是解三对角方程组的优秀算法.

下面给出追赶法的 MATLAB 程序.

- 追赶法 MATLAB 程序

```
%machase.m
function x= machase(a,b,c,d)
%用途: 追赶法解三对角方程组Ax=d
%格式: x= machase(a,b,c,d)  a为次下对角线元素向量, b主对角
%元素向量, c为次上对角线元素向量, d为右端向量,x返回解向量
n=length(a);
for k=2:n
  b(k)=b(k)-a(k)/b(k-1)*c(k-1);
  d(k)=d(k)-a(k)/b(k-1)*d(k-1);
end
x(n)=d(n)/b(n);
for k=n-1:-1:1
  x(k)=(d(k)-c(k)*x(k+1))/b(k);
end
```

例 3.6 用追赶法通用程序 machase.m 计算下列三对角方程组的解

$$\begin{pmatrix} 4 & 1 & & & & \\ 1 & 4 & 1 & & & \\ & 1 & 4 & 1 & & \\ & & 1 & \ddots & \ddots & \\ & & & \ddots & 4 & 1 \\ & & & & 1 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{49} \\ x_{50} \end{pmatrix} = \begin{pmatrix} 5 \\ 6 \\ 6 \\ \vdots \\ 6 \\ 5 \end{pmatrix}.$$

解 在 MATLAB 命令窗口执行:

```
>> a=ones(50,1); b=4*ones(50,1); c=ones(50,1);
>> d=6*ones(50,1); d(1)=5; d(50)=5;
>> x=machase(a,b,c,d)
```

得到计算结果：$x = (x_1, x_2, \cdots, x_{50}) = (1.0, 1.0, \cdots, 1.0)$.

例 3.7 *若方程组 (3.7) 的系数矩阵的元素满足条件：*

$$|b_i| > |c_i| > 0,\ |b_{i+1}| > |a_{i+1}| > 0,\ \ i = 1, \cdots, n-1,$$

则追赶法是可行的.

证 由 (3.8)~(3.9) 可知, 只需证明 $\bar{b}_k \neq 0$ $(k = 1, 2, \cdots, n)$ 即可. 显然 $\bar{b}_1 = b_1 \neq 0$. 当 $k \geqslant 2$ 时, 有

$$|\bar{b}_k| = \left| b_k - \frac{a_k}{\bar{b}_{k-1}} c_{k-1} \right| \geqslant |b_k| - |a_k| \cdot \left| \frac{c_{k-1}}{\bar{b}_{k-1}} \right| > |b_k| - |a_k| > 0,$$

即 $\bar{b}_k \neq 0$, $k = 2, \cdots, n$. 从而, 追赶法是可行的. □

3.4 LU 分解法

3.4.1 算法原理及其程序实现

首先看一个例题.

例 3.8 *设 $A \in \mathbf{R}^{n\times n}$ 非奇异, 若其顺序主子式 D_i $(i = 1, \cdots, n-1)$ 均不为零, 则存在唯一的单位下三角矩阵 L 和上三角矩阵 U, 使得 $A = LU$.*

证 由例 3.3 可知, 顺序 Gauss 消去法是可行的. 从消去法的过程, 可得

$$L_{n-1}^{-1} \cdots L_2^{-1} L_1^{-1} A = U,$$

其中 U 是一个上三角矩阵, L_i^{-1} 是对角线元素为 1, 第 i 对角线元素以下的元素为 $-m_{ik}(= -a_{ik}^{(k)}/a_{kk}^{(k)})$, 其余元素全为零的单位下三角矩阵 $(i = 1, \cdots, n-1,\ k = i+1, \cdots, n)$. 由此可得

$$A = L_1 L_2 \cdots L_{n-1} U.$$

令 $L = L_1 L_2 \cdots L_{n-1}$, 则 L 是一个单位下三角矩阵. 从而有 $A = LU$. □

现在我们来讨论矩阵的 LU 分解. 设 $A = LU$, 其中 L 为一个单位下三角矩阵, U 为一个上三角矩阵, 即

$$L = \begin{pmatrix} 1 & & & \\ l_{21} & 1 & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \cdots & 1 \end{pmatrix},\quad U = \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ & u_{22} & \cdots & u_{2n} \\ & & \ddots & \vdots \\ & & & u_{nn} \end{pmatrix}, \tag{3.10}$$

则称 $A=LU$ 为一个 LU 分解. 这时线性方程组

$$Ax=b \quad \Rightarrow \quad LUx=b \quad \Rightarrow \quad \begin{cases} Ly=b, \\ Ux=y \end{cases} \tag{3.11}$$

转化为 $Ly=b$ 及 $Ux=y$ 两个三角形方程组. 由于三角形方程组很容易通过回代方法求解, 只有 $O(n^2)$ 的计算量.

下面我们来推导 LU 分解的计算公式. 由等式 $A=LU$, 可得

$$a_{ij}=(l_{i1},\cdots,l_{i,i-1},1,0,\cdots,0)\times(u_{1j},\cdots,u_{jj},0,\cdots,0)^{\mathrm{T}}. \tag{3.12}$$

当 $j \geqslant i$ 时,

$$a_{ij}=l_{i1}u_{1j}+\cdots+l_{i,i-1}u_{i-1,j}+u_{ij},$$

于是

$$u_{ij}=a_{ij}-\sum_{r=1}^{i-1}l_{ir}u_{rj};$$

当 $j<i$ 时,

$$a_{ij}=l_{i1}u_{1j}+\cdots+l_{i,j-1}u_{j-1,j}+l_{ij}u_{jj},$$

于是

$$l_{ij}=\left(a_{ij}-\sum_{r=1}^{j-1}l_{ir}u_{rj}\right)/u_{jj}.$$

即

$$u_{1j}=a_{1j},\ \ j=1,\cdots,n;\ \ l_{i1}=a_{i1}/u_{11},\ \ i=2,\cdots,n; \tag{3.13}$$

$$u_{ij}=a_{ij}-\sum_{r=1}^{i-1}l_{ir}u_{rj},\ \ i=2,\cdots,n;\ j=i,\cdots,n; \tag{3.14}$$

$$l_{ij}=\left(a_{ij}-\sum_{r=1}^{j-1}l_{ir}u_{rj}\right)/u_{jj},\ \ i=2,\cdots,n;\ j=2,\cdots,i-1. \tag{3.15}$$

为了便于编程计算, 将 (3.14) 中的下标 i 换成 k, 将 (3.15) 中的下标 j 换成 k, 则有

$$u_{kj}=a_{kj}-\sum_{r=1}^{k-1}l_{kr}u_{rj},\ \ k=2,\cdots,n;\ j=k,\cdots,n; \tag{3.16}$$

$$l_{ik}=\left(a_{ik}-\sum_{r=1}^{k-1}l_{ir}u_{rk}\right)/u_{kk},\ \ k=2,\cdots,n-1;\ i=k+1,\cdots,n. \tag{3.17}$$

下面是用 LU 分解求解线性方程组的算法步骤:

算法 3.3 (LU分解法)

步 1 输入系数矩阵 A, 右端项 b;

步 2 LU 分解:

$u_{1j} = a_{1j}, \quad j = 1, \cdots, n;$

$l_{i1} = a_{i1}/u_{11}, \quad i = 2, \cdots, n;$

对 $k = 2, \cdots, n$, 计算

$$u_{kj} = a_{kj} - \sum_{r=1}^{k-1} l_{kr} u_{rj}, \quad j = k, \cdots, n;$$

$$l_{ik} = \left(a_{ik} - \sum_{r=1}^{k-1} l_{ir} u_{rk}\right) / u_{kk}, \quad i = k+1, \cdots, n.$$

步 3 用向前消去法解下三角方程组 $Ly = b$:

$y_1 = b_1$,

对 $k = 2, \cdots, n$, 计算

$$y_k = b_k - \sum_{j=1}^{k-1} l_{kj} y_j;$$

步 4 用回代法解上三角方程组 $Ux = y$:

$x_n = y_n / u_{nn}$,

对 $k = n-1, \cdots, 1$, 计算

$$x_k = \left(y_k - \sum_{j=k+1}^{n} u_{kj} x_j\right) / u_{kk}.$$

根据算法 3.3, 编制 MATLAB 程序如下:

- LU 分解 MATLAB 程序

```
%malu.m
function [x,l,u]=malu(A,b)
%用途: 用LU分解法解方程组Ax=b
%格式: [x,l,u]=malu(A,b)  A为系数矩阵, b为右端向量,
%x返回解向量, l返回下三角矩阵, u返回上三角矩阵
%LU 分解
n=length(b); u=zeros(n,n);
l=eye(n,n);  u(1,:)=A(1,:);
l(2:n,1)=A(2:n,1)/u(1,1);
for k=2:n
```

```
    u(k,k:n)=A(k,k:n)-l(k,1:k-1)*u(1:k-1,k:n);
    l(k+1:n,k)=(A(k+1:n,k)-l(k+1:n,1:k-1)*u(1:k-1,k))/u(k,k);
end
%解下三角方程组Ly=b
y=zeros(n,1);
y(1)=b(1);
for k=2:n
    y(k)=b(k)-l(k,1:k-1)*y(1:k-1);
end
%解上三角方程组Ux=y
x=zeros(n,1);
x(n)=y(n)/u(n,n);
for k=n-1:-1:1
    x(k)=(y(k)-u(k,k+1:n)*x(k+1:n))/u(k,k);
end
```

例 3.9 利用程序 malu.m 计算下列线性方程组的解

$$\begin{pmatrix} 2 & -1 & 4 & -3 & 1 \\ -1 & 1 & 2 & 1 & 3 \\ 4 & 2 & 3 & 3 & -1 \\ -3 & 1 & 3 & 2 & 4 \\ 1 & 3 & -1 & 4 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 11 \\ 14 \\ 4 \\ 16 \\ 18 \end{pmatrix}.$$

解 在 MATLAB 命令窗口执行

```
>> A=[2 -1 4 -3 1;-1 1 2 1 3;4 2 3 3 -1;-3 1 3 2 4;1 3 -1 4 4];
>> b=[11 14 4 16 18]';
>> [x,l,u]=malu(A,b); x'
```

得计算结果:

```
x =
    1.0000    2.0000    1.0000   -1.0000    4.0000
l =
    1.0000         0         0         0         0
   -0.5000    1.0000         0         0         0
    2.0000    8.0000    1.0000         0         0
   -1.5000   -1.0000   -0.3514    1.0000         0
    0.5000    7.0000    0.8378   -1.2069    1.0000

u =
```

```
2.0000   -1.0000    4.0000   -3.0000    1.0000
     0    0.5000    4.0000   -0.5000    3.5000
     0         0  -37.0000   13.0000  -31.0000
     0         0         0    1.5676   -1.8919
     0         0         0         0    2.6897
```

3.4.2 LU 分解与 Gauss 消去法的关系

现在我们来讨论 LU 分解与 Gauss 消去法的关系. 由算法 3.1 可知, 顺序 Gauss 消去法的第一步消元相当于用矩阵

$$M_1 = \begin{pmatrix} 1 & & & & \\ -m_{21} & 1 & & & \\ -m_{31} & & 1 & & \\ \vdots & & & \ddots & \\ -m_{n1} & & & & 1 \end{pmatrix}$$

左乘 $(A^{(1)}, b^{(1)})$, 这里 m_{i1} 由 (3.4) 所定义, 即

$$(A^{(2)}, b^{(2)}) = M_1(A^{(1)}, b^{(1)}).$$

第二步消元相当于用矩阵

$$M_2 = \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & -m_{32} & 1 & & \\ & \vdots & & \ddots & \\ & -m_{n2} & & & 1 \end{pmatrix}$$

左乘 $(A^{(2)}, b^{(2)})$, 即

$$(A^{(3)}, b^{(3)}) = M_2(A^{(2)}, b^{(2)}) = M_2 M_1(A^{(1)}, b^{(1)}).$$

一般地, 第 k 步相当于用矩阵

$$M_k = \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & -m_{k+1,k} & 1 & & \\ & & \vdots & & \ddots & \\ & & -m_{nk} & & & 1 \end{pmatrix}$$

左乘 $(A^{(k)}, b^{(k)})$, 即

$$\begin{aligned}(A^{(k+1)}, b^{(k+1)}) &= M_k(A^{(k)}, b^{(k)}) = \cdots \\ &= M_k \cdots M_2 M_1 (A^{(1)}, b^{(1)}), \quad k = 1, \cdots, n-1.\end{aligned}$$

由 Gauss 消去法可知, 经过 $n-1$ 步消元后, 系数矩阵 A 被化成了上三角矩阵, 即 $A^{(n)} = U$, 从而

$$U = A^{(n)} = M_{n-1}A^{(n-1)} = M_{n-1} \cdots M_2 M_1 A,$$

于是

$$A = M_1^{-1} M_2^{-1} \cdots M_{n-1}^{-1} U = LU,$$

这里,

$$L = M_1^{-1} M_2^{-1} \cdots M_{n-1}^{-1} = \begin{pmatrix} 1 & & & & \\ m_{21} & 1 & & & \\ \vdots & & \ddots & & \\ m_{n-1,1} & m_{n-1\,2} & \cdots & 1 & \\ m_{n1} & m_{n2} & \cdots & m_{n,n-1} & 1 \end{pmatrix}$$

是单位下三角矩阵. 由此可见, 顺序 Gauss 消去法实际上就是将方程组的系数矩阵分解成单位下三角矩阵与上三角矩阵的乘积. 对比算法 3.1 和算法 3.3, 不难看出, 顺序 Gauss 消去法的消元过程相当于 LU 分解过程和 $Ly = b$ 的求解, 而回代过程则相当于解线性方程组 $Ux = y$.

3.5 解对称正定方程组的 Cholesky 分解法

对称正定方程组在工程计算中有着广泛而重要的应用. 当方程组的系数矩阵 A 为对称正定时, 存在一个实的非奇异下三角矩阵 L 使

$$A = LL^{\mathrm{T}}, \tag{3.18}$$

且当限定 L 的对角线元素为正时, 这种分解是唯一的.

设

$$L = \begin{pmatrix} l_{11} & & & \\ l_{21} & l_{22} & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{pmatrix},$$

由 $A=LL^{\mathrm{T}}$ 比较 A 和 LL^{T} 的对应元素, 可求得 L 的元素 l_{ij} 如下:

由 $a_{11}=l_{11}^2$, $a_{i1}=l_{i1}l_{11}$, 得

$$
\begin{aligned}
&l_{11}=\sqrt{a_{11}},\\
&l_{i1}=a_{i1}/l_{11},\quad i=2,\cdots,n.
\end{aligned}
$$

假设 L 的第 $k-1$ 列元素已经求得, 下面求 L 的第 k 列元素 l_{ik}, $i=k,\cdots,n$. 注意到

$$
a_{ik}=\sum_{r=1}^{k-1}l_{ir}l_{kr}+l_{ik}l_{kk},
$$

得

$$
\begin{aligned}
&l_{kk}=\left(a_{kk}-\sum_{r=1}^{k-1}l_{kr}^2\right)^{1/2},\\
&l_{ik}=\left(a_{ik}-\sum_{r=1}^{k-1}l_{ir}l_{kr}\right)\Big/l_{kk},\quad i=k+1,\cdots,n.
\end{aligned}
\tag{3.19}
$$

上述分解方法称为 Cholesky 分解法. 由于计算的对角线元素需要作 n 次开平方运算, 故 Cholesky 分解法又称为平方根法. 不难验证, Cholesky 分解法的乘除计算总量约为 $n^3/6+O(n^2)$, 为一般矩阵 LU 分解计算量的一半. 虽然如此, 但其增加的 n 个开方运算是非常不利的.

为了避免开方运算, 把矩阵 A 分解成

$$
A=LDL^{\mathrm{T}},\tag{3.20}
$$

其中 L 为单位下三角矩阵, D 是对角矩阵, 且对角元均不为零. 利用 (3.20) 两边元素对应相等的办法, 可得出计算 L 和 D 的计算公式. 假设 L 和 D 的第 1 至 $k-1$ 列元素已经求得, 以下求它们的第 k 列元素 l_{ik}, $i=k+1,\cdots,n$ 和 d_k, 这里 $l_{kk}=1$. 比较 $A=LDL^{\mathrm{T}}$ 两边的第 k 列, 得

$$
a_{ik}=\sum_{r=1}^{k-1}l_{ir}d_rl_{kr}+l_{ik}d_k,\quad i=k,\cdots,n.
$$

由此得

$$
\begin{aligned}
&d_k=a_{kk}-\sum_{r=1}^{k-1}l_{kr}^2d_r,\\
&l_{ik}=\left(a_{ik}-\sum_{r=1}^{k-1}l_{ir}d_rl_{kr}\right)\Big/d_k,\quad i=k+1,\cdots,n.
\end{aligned}
\tag{3.21}
$$

按 (3.21) 计算, LDL^{T} 分解可避免开方运算, 但由计算公式不难发现, 乘除运算总量增加了一倍, 又恢复到 $n^3/3+O(n^2)$.

为了减少乘法运算量, 引入辅助变量 $t_{ik}=l_{ik}d_k$, 并将计算公式 (3.21) 整理如下:

对 $k=1,2,\cdots,n$, 计算

$$\begin{aligned}
&t_{ik}=a_{ik}-\sum_{j=1}^{k-1}t_{ij}l_{kj},\quad i=k+1,\cdots,n,\\
&l_{ik}=t_{ik}/d_k,\quad i=k+1,\cdots,n,\\
&d_k=a_{kk}-\sum_{j=1}^{k-1}t_{kj}l_{kj}.
\end{aligned}\tag{3.22}$$

容易看出, 改进后的 LDL^{T} 分解乘除运算量约为 $n^3/6+O(n^2)$, 不需要开方运算. 但该算法存储变量 t_{ik}, 存储量几乎增加一倍.

下面我们建立用 Cholesky 分解法求解对称正定方程组的算法步骤.

$$Ax=b\quad\Rightarrow\quad LDL^{\mathrm{T}}x=b\quad\Rightarrow\quad\begin{cases}Ly=b,\\Dz=y,\\L^{\mathrm{T}}x=z.\end{cases}\tag{3.23}$$

算法 3.4 (Cholesky 分解法)

步 1 输入对称正定矩阵 A 和右端向量 b;

步 2 Cholesky 分解:

$d_1=t_{11}=a_{11},\ \ l_{i1}=a_{i1}/d_1,\ i=2,\cdots,n,$

对 $k=2,\cdots,n$ 计算:

$$d_k=a_{kk}-\sum_{j=1}^{k-1}t_{kj}l_{kj},$$

$$t_{ik}=a_{ik}-\sum_{j=1}^{k-1}t_{ij}l_{kj},\quad l_{ik}=t_{ik}/d_k,\ i=k+1,\cdots,n;$$

步 3 用向前消去法解下三角方程组 $Ly=b$:

$y_1=b_1,$

对 $k=2,\cdots,n$ 计算 $y_k=b_k-\sum_{j=1}^{k-1}l_{kj}y_j$;

步 4 解对角形方程组 $Dz=y$:

对 $k=1,\cdots,n$, 计算: $z_k=y_k/d_k$;

步 5 用回代法解上三角方程组 $L^{\mathrm{T}}x = z$:

$$x_n = z_n,$$

$$\text{对 } k = n-1, \cdots, 1 \text{ 计算:} \quad x_k = z_k - \sum_{j=k+1}^{n} l_{jk} x_j.$$

根据算法 3.4, 编制 MATLAB 程序如下:

- Cholesky 分解法 MATLAB 程序

```
%machol.m
function [x,l,d]=machol(A,b)
%用途: 用 Cholesky分解法解方程组Ax=b
%LDL'分解
n=length(b); d=zeros(1,n);
l=eye(n,n);
d(1)=A(1,1);
l(2:n,1)=A(2:n,1)/d(1);
d(2)=A(2,2)-l(2,1)*l(2,1)*d(1);
for i=3:n
   for j=2:(i-1)
      s=0;
      for k=1:(j-1) s=s+d(k)*l(i,k)*l(j,k); end
      l(i,j)=(A(i,j)-s)/d(j);
   end
   s=0;
   for j=1:(i-1)  s=s+d(j)*l(i,j)*l(i,j); end
   d(i)=A(i,i)-s;
end
%求解下三角方程组Ly=b(向前消去法)
y=zeros(n,1);
y(1)=b(1);
for i=2:n
    y(i)=b(i)-l(i,1:i-1)*y([1:i-1]);
end
%求解对角方程组Dz=y
for i=1:n  z(i)=y(i)/d(i); end
%求解上三角方程组L'x=z(回代法)
```

```
ll=l'; x=zeros(n,1); x(n)=z(n);
for i=(n-1):-1:1
    x(i)=z(i)-ll(i,i+1:n)*x(i+1:n);
end
x=x';
```

例 3.10 利用程序 machol.m 计算下列线性方程组的解

$$\begin{pmatrix} 2 & -1 & 4 & -3 & 1 \\ -1 & 1 & 2 & 1 & 3 \\ 4 & 2 & 3 & 3 & -1 \\ -3 & 1 & 3 & 2 & 4 \\ 1 & 3 & -1 & 4 & 4 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix}=\begin{pmatrix} 11 \\ 14 \\ 4 \\ 16 \\ 18 \end{pmatrix}$$

解 在 MATLAB 命令窗口执行

```
>> A=[2 -1 4 -3 1;-1 1 2 1 3;4 2 3 3 -1;-3 1 3 2 4;1 3 -1 4 4];
>> b=[11 14 4 16 18]';
>> [x,l,d]=machol(A,b)
```

得计算结果:

```
x =
    1.0000    2.0000    1.0000   -1.0000    4.0000
l =
    1.0000         0         0         0         0
   -0.5000    1.0000         0         0         0
    2.0000    8.0000    1.0000         0         0
   -1.5000   -1.0000   -0.3514    1.0000         0
    0.5000    7.0000    0.8378   -1.2069    1.0000
d =
    2.0000    0.5000  -37.0000    1.5676    2.6897
```

例 3.11 已知方程组

$$\begin{pmatrix} 2 & -1 & b \\ -1 & 2 & a \\ b & -1 & 2 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}=\begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix}.$$

试问参数 a, b 满足什么条件时, 可选用 Cholesky 分解法求解该方程组?

解 方程组系数矩阵 A 对称正定时, 可用 Cholesky 分解法求解. 由 $A^{\mathrm{T}}=A$ 可得 $a=-1$. 对称矩阵正定的充分必要条件是其各阶顺序主子式均大于零. 注意

到 $D_1 = 2 > 0$, $D_2 = 4 - 1 = 3 > 0$. 而由

$$D_3 = \begin{vmatrix} 2 & -1 & b \\ -1 & 2 & -1 \\ b & -1 & 2 \end{vmatrix} = 4 - 2b - 2b^2 > 0,$$

可得 $-1 < b < 2$. 故当 $a = -1$, $-1 < b < 2$ 时, 上述方程组可用 Cholesky 分解法来求解.

3.6 舍入误差对解的影响

用直接法解线性方程组 $Ax = b\,(\det(A) \neq 0)$, 理应得出准确解 x. 但因为存在舍入误差, 只能得出近似解 $\bar{x}$, 或者说得到近似方程组 $\bar{A}\bar{x} = \bar{b}$ 的准确解. 近似矩阵 $\bar{A}$ 和近似向量 $\bar{b}$ 的误差

$$\delta A = A - \bar{A}, \quad \delta b = b - \bar{b}$$

同计算机运算和精度有关. 计算精度越高, $\|\delta A\|$ 和 $\|\delta b\|$ 必然越小. 下面估计 $\|\delta A\|$ 和 $\|\delta b\|$ 很小时解的误差 $\delta x = x - \bar{x}$. 注意到 x 和 $\bar{x}$ 分别满足方程组

$$Ax = b, \quad (A - \delta A)(x - \delta x) = b - \delta b.$$

两式相减得

$$(A - \delta A)\delta x = \delta b - \delta A x.$$

当 $\|\delta A\|$ 很小时, $\|A^{-1}\delta A\|$ 也很小, $A - \delta A = A(I - A^{-1}\delta A)$ 可逆, 于是

$$\delta x = (A - \delta A)^{-1}(\delta b - \delta A x) = (I - A^{-1}\delta A)^{-1}A^{-1}(\delta b - \delta A x).$$

故

$$\begin{aligned} \|\delta x\| &= \|(I - A^{-1}\delta A)^{-1}A^{-1}(\delta b - \delta A x)\| \\ &\leqslant \|(I - A^{-1}\delta A)^{-1}\| \cdot \|A^{-1}\|(\|\delta b\| + \|\delta A x\|) \\ &\leqslant \frac{\|A^{-1}\|}{1 - \|A^{-1}\delta A\|}(\|\delta b\| + \|\delta A\| \cdot \|x\|). \end{aligned}$$

上面的最后一个不等式用到了定理 2.3. 注意到,

$$\|A^{-1}\delta A\| \leqslant \|A^{-1}\| \cdot \|\delta A\|, \quad \|b\| = \|Ax\| \leqslant \|A\| \cdot \|x\|,$$

从而有

$$\frac{\|\delta x\|}{\|x\|} \leqslant \frac{\|A^{-1}\|}{1-\|A^{-1}\|\cdot\|\delta A\|}\left(\frac{\|A\|\cdot\|\delta b\|}{\|A\|\cdot\|x\|}+\frac{\|A\|\cdot\|\delta A\|}{\|A\|}\right)$$
$$\leqslant \frac{\|A^{-1}\|\cdot\|A\|}{1-\|A^{-1}\|\cdot\|A\|\dfrac{\|\delta A\|}{\|A\|}}\left(\frac{\|\delta b\|}{\|b\|}+\frac{\|\delta A\|}{\|A\|}\right).$$

令 $\kappa=\mathrm{Cond}(A)=\|A^{-1}\|\cdot\|A\|$, 则得近似解 $\bar{x}$ 的相对误差估计式

$$\frac{\|\delta x\|}{\|x\|} \leqslant \frac{\kappa}{1-\kappa\cdot\varepsilon_r(\bar{A})}[\varepsilon_r(\bar{b})+\varepsilon_r(\bar{A})], \tag{3.24}$$

其中

$$\varepsilon_r(\bar{A})=\frac{\|\delta A\|}{\|A\|}, \quad \varepsilon_r(\bar{b})=\frac{\|\delta b\|}{\|b\|}.$$

上式表明, 当 $\varepsilon_r(\bar{A})=\|\delta A\|/\|A\|$ 很小时, 解的相对误差约等于 $\bar{A}$ 和 $\bar{b}$ 的相对误差的 κ 倍; 而 κ 当很大时, 即使 $\bar{A}$ 和 $\bar{b}$ 的相对误差很小, 解的相对误差也可能很大. 由此可知, 舍入误差对解的影响的大小取决于数 κ 的大小, 我们把这个数称为方程组的条件数. 条件数 κ 很大的方程组称为病态方程组, κ 较小的方程组称为良态方程组.

对于病态方程组, 为了得到较准确的近似解, 可以采用以下措施来减少舍入误差的影响: (1) 采用高精度计算; (2) 采用数值稳定性较好的算法, 如全主元 Gauss 消去法等; (3) 采用迭代改善计算解的办法.

所谓迭代改善计算解 $\bar{x}$, 目的是设法求取修正量 Δx, 使 $\bar{x}+\Delta x$ 满足原方程组 $Ax=b$, 即

$$A(\bar{x}+\Delta x)=b, \quad A\Delta x=r=b-A\bar{x}.$$

实际计算时, 方程组 $A\Delta x=r$ 不大可能准确求解, 从而必须反复求解 $A\Delta x=r$ 和修正 $\bar{x}$, 使 $\bar{x}$ 逐渐接近真解. 这一过程称为迭代改善. 为节省计算量, 最好事先将系数矩阵 A 进行 LU 分解: $A=LU$, 反复求解 $A\Delta x=r$ 改为反复求解 $Ly=r$ 和 $U\Delta x=y$. 为保证计算精度, 计算残矢量 r 最好采用高精度计算. 迭代改善过程可表述如下:

(1) LU 分解: $A=LU$;

(2) 高精度计算: $r=b-A\bar{x}$;

(3) 求解: $Ly=r$ 和 $U\Delta x=y$, 置 $\bar{x}:=\bar{x}+\Delta x$;

(4) 当 $\|\Delta x\|$ 很小时, 停算, 否则, 转 (2).

例 3.12　已知方程组

$$\begin{pmatrix} 1 & 0 & -1 \\ 2 & 2 & 1 \\ 0 & 2 & 2 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}=\begin{pmatrix} -1 \\ 2 \\ 2 \end{pmatrix}$$

的解为 $x=(1,\ -1,\ 2)^{\mathrm{T}}$. 如果右端有微小扰动 $\|\delta b\|_\infty=0.5\times10^{-6}$, 估计由此引起的解的相对误差.

解　记方程组的系数矩阵为 A. 由于

$$A^{-1}=\begin{pmatrix} -1 & 1 & -1 \\ 2 & -1 & 1.5 \\ -2 & 1 & -1 \end{pmatrix},$$

从而 $\mathrm{Cond}(A)_\infty=\|A\|_\infty\|A^{-1}\|_\infty=5\times4.5=22.5$. 故由公式

$$\frac{\|\delta x\|_\infty}{\|x\|_\infty}\leqslant \mathrm{Cond}(A)_\infty\frac{\|\delta b\|_\infty}{\|b\|_\infty}$$

可得

$$\frac{\|\delta x\|_\infty}{\|x\|_\infty}\leqslant 22.5\times\frac{0.5\times10^{-6}}{2}=5.625\times10^{-6}.$$

由上述结果可以看出, 解的相对误差是右端扰动量的 11 倍多.

习　题　3

(Ⅰ) 理论分析题

3.1　用列主元 Gauss 消去法解下面的方程组

(1) $\begin{cases} -3x_1+2x_2+6x_3=4, \\ 10x_1-7x_2+2x_3=9, \\ -5x_1-2x_2+5x_3=7, \end{cases}$　　(2) $\begin{cases} x_1+2x_2+3x_3=1, \\ 5x_1+4x_2+10x_3=0, \\ 3x_1-0.1x_2+x_3=2. \end{cases}$

3.2　设 A 为 n 阶矩阵, 对线性方程组 $Ax=b$ 估计顺序 Gauss 消去法的乘除运算总量.

3.3　顺序 Gauss 消去法可行的条件是 $a_{11}^{(1)},a_{22}^{(2)},\cdots,a_{n-1,n-1}^{(n-1)}$ 都不为零. 试证明顺序 Gauss 消去法可行的充要条件是 A 的顺序主子式 $D_k\neq0,\ 1\leqslant k\leqslant n$.

3.4　设 $A=(a_{ij})$, $a_{11}\neq0$, 经一步 Gauss 消去后得到

$$A^{(2)}=\begin{pmatrix} a_{11} & \alpha_1^{\mathrm{T}} \\ O & A_2 \end{pmatrix},\quad 其中\ A_2=\begin{pmatrix} a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & & \vdots \\ a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} \end{pmatrix},$$

试证明：(1) 若 A 对称，则 A_2 也对称；(2) 若 A 对称正定，则 A_2 也对称正定.

3.5 对于 n 阶矩阵 $A=(a_{ij})$，若

$$|a_{ii}|>\sum_{j=1,j\neq i}^{n}|a_{ij}|,\quad i=1,2,\cdots,n,$$

则称 A 是严格对角占优矩阵. 证明：若 A 是严格对角占优矩阵，则经一步顺序 Gauss 消元过程后，得到的 $A^{(1)}$ 仍为严格对角占优矩阵.

3.6 已知方程组 $Ax=f$，其中

$$A=\begin{pmatrix}2 & -1 & b\\ -1 & 2 & a\\ b & -1 & 2\end{pmatrix},\quad f=\begin{pmatrix}0\\1\\0\end{pmatrix}.$$

(1) 试问参数和满足什么条件式时，可选用平方根法求解该方程组？

(2) 取 $b=0$，$a=1$，试用追赶法求解该方程组.

3.7 证明：(1) 正定矩阵必存在 LU 分解；(2) 如果对称矩阵的各阶顺序主子式不等于零，则必存在 LU 分解.

3.8 证明：非奇异矩阵 A 不一定有 LU 分解.

3.9 证明：非奇异矩阵 $A\in\mathbf{R}^{n\times n}$ 有唯一 LDU 分解的充要条件是 A 的顺序主子式 $D_1,D_2,\cdots,D_{n-1}$ 都是非零的，其中 D 是对角矩阵，L，U 分别是单位下三角和单位上三角矩阵.

3.10 设 U 为非奇异的上三角矩阵.

(1) 推导求解 $Ux=d$ 的一般公式，并写出算法；

(2) 计算求解上三角形方程组 $Ux=d$ 的乘除法次数.

3.11 设 L 为非奇异的下三角矩阵.

(1) 列出逐次代入求解 $Lx=d$ 的公式；

(2) 上述求解过程共需多少次乘除法运算？

(3) 给出求 L^{-1} 的计算公式.

3.12 试证明：

(1) 如果 A 是对称正定矩阵，则 A 可以唯一地写成 $A=LL^{\mathrm{T}}$，其中 L 是具有正对角元素的下三角矩阵.

(2) 如果 A 是对称正定矩阵，则 A^{-1} 也是对称正定矩阵.

3.13 求下列矩阵的条件数

$$A=\begin{pmatrix}1 & 0\\ 0 & 10^{-10}\end{pmatrix}.$$

3.14 方程组 $Ax=b$, 其中 A 为 $m\times n$ 阶对称且非奇异矩阵. 设 A 有误差 δA, 则原方程组变化为 $(A+\delta A)(x+\delta x)b$, 其中 δx 为解的误差向量. 证明

$$\frac{\|\delta x\|_2}{\|x+\delta x\|_2}\leqslant\left|\frac{\lambda_1}{\lambda_n}\right|\frac{\|\delta A\|_2}{\|A\|_2},$$

其中 λ_1 和 λ_n 分别为 A 的按模最大和最小的特征值.

3.15 设扰动方程组为 $(A+\delta A)(x+\delta x)=b+\delta b$, 证明：当 $1-\|A^{-1}\|\cdot\|\delta A\|>0$ 时, 有

(1) $\|\delta x\|\leqslant\dfrac{\|A^{-1}\|}{1-\|A^{-1}\|\cdot\|\delta A\|}(\|\delta b\|+\|\delta A\|\cdot\|x\|)$;

(2) $\dfrac{\|\delta x\|}{\|x\|}\leqslant\dfrac{\|A^{-1}\|\cdot\|A\|}{1-\|A^{-1}\|\cdot\|A\|\dfrac{\|\delta A\|}{\|A\|}}\left(\dfrac{\|\delta b\|}{\|b\|}+\dfrac{\|\delta A\|}{\|A\|}\right)$.

(Ⅱ) 上机实验题

3.1 利用算法 3.1 (顺序 Gauss 消去法), 编制 MATLAB 程序, 求线性方程组

(1) $\begin{pmatrix}2&3&4&5\\3&5&2&1\\4&3&12&5\\5&6&7&8\end{pmatrix}\begin{pmatrix}x_1\\x_2\\x_3\\x_4\end{pmatrix}=\begin{pmatrix}24\\-5\\34\\33\end{pmatrix}$;

(2) $\begin{pmatrix}10.4&1.2&2.2&1.9\\1.5&11.2&3.5&2.5\\2.1&1.5&9.6&1.8\\1.6&4.5&1.4&12.8\end{pmatrix}\begin{pmatrix}x_1\\x_2\\x_3\\x_4\end{pmatrix}=\begin{pmatrix}10.54\\-22.47\\-18.27\\29.93\end{pmatrix}$

的近似解.

3.2 利用算法 3.2 (列主元 Gauss 消去法), 编制 MATLAB 程序, 求下列方程组

(1) $\begin{cases}12x_1-2x_2+3x_3=15,\\18x_1+3x_2-2x_3=-12,\\-x_1-x_2-15x_3=21;\end{cases}$

(2) $\begin{pmatrix}3&-1&4\\-1&2&-2\\2&-3&-2\end{pmatrix}\begin{pmatrix}x_1\\x_2\\x_3\end{pmatrix}=\begin{pmatrix}5\\2\\7\end{pmatrix}$

的近似解.

3.3 利用追赶法, 编制 MATLAB 程序, 求下列三对角方程组

$$\begin{pmatrix} 4 & 1 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ 0 & 0 & 1 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 9 \\ 10 \\ 20 \\ 16 \end{pmatrix}$$

的近似解.

3.4 利用算法 3.3 (LU 分解法), 编制 MATLAB 程序, 求下列方程组

$$(1) \begin{cases} x_1 + 2x_2 + 3x_3 = 1, \\ 5x_1 + 4x_2 + 10x_3 = 0, \\ 3x_1 + 0.2x_2 + 2x_3 = 2; \end{cases}$$

$$(2) \begin{cases} 2x_1 + 2x_2 + 3x_3 = 7, \\ 4x_1 + 7x_2 + 7x_3 = 18, \\ -2x_1 + 4x_2 + 5x_3 = 1 \end{cases}$$

的近似解.

3.5 利用算法 3.4 (Cholesky 分解法), 编制 MATLAB 程序, 求方程组

$$(1) \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 2 & 2 \\ 1 & 2 & 3 & 3 \\ 1 & 2 & 3 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 4 \\ 3 \\ 2 \\ 1 \end{pmatrix};$$

$$(2) \begin{pmatrix} 3 & 3 & 5 & 7 \\ 3 & 5 & 7 & 9 \\ 5 & 7 & 9 & 3 \\ 7 & 9 & 3 & 11 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 6 \\ 8 \\ 7 \\ 9 \end{pmatrix}$$

的近似解.

第 4 章　插值法与最小二乘拟合

已知函数 $y=f(x)$ 的一批数据 $(x_1,y_1),(x_2,y_2),\cdots,(x_n,y_n)$, 而函数的表达式未知, 要从某类函数 (如多项式函数、样条函数等) 中求得一个函数 $\varphi(x)$ 作为 $f(x)$ 的近似, 这类数值计算问题称为数据建模. 有时尽管 $y=f(x)$ 有表达式, 但比较复杂, 我们也利用该方法建立一个近似模型.

数据建模有两大类方法：一类是插值方法, 要求所求函数 $\varphi(x)$ 严格遵从数据 $(x_1,y_1),(x_2,y_2),\cdots,(x_n,y_n)$; 另一类是拟合方法, 允许函数 $\varphi(x)$ 在数据点上有误差, 但要求达到某种误差指标最小化. 其中, 以误差向量的 2- 范数为误差指标的数据拟合称为最小二乘拟合. 一般而言, 插值方法比较适合数据准确或数据量较小的情形, 而拟合方法则比较适合数据有误差或数据量较大的情形.

4.1　多项式插值

4.1.1　插值多项式的概念

在众多的函数中, 多项式最简单、最容易计算. 因此, 已知函数 $y=f(x)$ 在 $n+1$ 个互不相同的点处的函数值 $y_i=f(x_i)$, $i=0,1,\cdots,n$, 为求 $y=f(x)$ 的近似式, 首先考虑的自然应当是选取 n 次多项式

$$P_n(x)=a_0+a_1x+\cdots+a_nx^n, \tag{4.1}$$

使 $P_n(x)$ 满足条件

$$P_n(x_i)=y_i,\quad i=0,1,\cdots,n. \tag{4.2}$$

函数 $f(x)$ 称为被插函数, $P_n(x)$ 称为插值多项式, 条件 (4.2) 称为插值条件, $x_0,x_1,\cdots,x_n$ 称为插值节点. 这种求函数近似式的方法称为插值法.

满足插值条件 (4.2) 的插值多项式 $P_n(x)$ 是唯一存在的. 事实上, 插值条件 (4.2) 可看成未知数是 $a_0,a_1,\cdots,a_n$ 的线性方程组

$$\begin{cases} a_0+x_0a_1+\cdots+x_0^na_n=y_0, \\ a_0+x_1a_1+\cdots+x_1^na_n=y_1, \\ \cdots\cdots\cdots\cdots \\ a_0+x_na_1+\cdots+x_n^na_n=y_n. \end{cases} \tag{4.3}$$

因系数行列式为范德蒙德行列式

$$D = \begin{vmatrix} 1 & x_0 & \cdots & x_0^n \\ 1 & x_1 & \cdots & x_1^n \\ \vdots & & & \vdots \\ 1 & x_n & \cdots & x_n^n \end{vmatrix} = \prod_{0 \leqslant i < j \leqslant n} (x_j - x_i) \neq 0,$$

知 (4.3) 必有唯一解.

4.1.2 插值多项式的截断误差

可以证明, 如果被插函数 $y = f(x)$ 在包含插值节点 $x_0, x_1, \cdots, x_n$ 的区间 $[a, b]$ 上存在 $n+1$ 阶导数, 则在区间 $[a, b]$ 任意点 x 处, 被插函数 $f(x)$ 与插值多项式 $P_n(x)$ 的截断误差为 (利用泰勒公式):

$$R_n(x) = f(x) - P_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\omega(x), \tag{4.4}$$

其中

$$\omega(x) = (x - x_0)(x - x_1)\cdots(x - x_n) = \prod_{j=0}^{n}(x - x_j), \tag{4.5}$$

ξ 介于 x 与节点 $x_0, x_1, \cdots, x_n$ 之间.

事实上, 当 x 为节点时, (4.4) 两边皆为零, 等式显然成立. 下面假定 x 不是节点. 作辅助函数

$$\varphi(t) = R_n(t) - \frac{R_n(x)}{\omega(x)}\omega(t).$$

不难发现

$$\varphi(x) = R_n(x) - \frac{R_n(x)}{\omega(x)}\omega(x) = 0,$$

$$\varphi(x_i) = f(x_i) - P_n(x_i) - \frac{R_n(x)}{\omega(x)}\omega(x_i) = 0, \quad i = 0, 1, \cdots, n,$$

即 $\varphi(t)$ 存在 $n+2$ 个零点 $x, x_0, x_1, \cdots, x_n$. 由微分学的罗尔中值定理知 $\varphi'(t)$ 存在 $n+1$ 个零点. 同样, 对 $\varphi'(t)$ 使用罗尔定理, 知 $\varphi''(t)$ 存在 n 个零点. 依此递推最后得 $\varphi^{(n+1)}(t)$ 存在 1 个零点, 记为 ξ (介于 x 与 $x_0, x_1, \cdots, x_n$ 之间). 直接计算得

$$\begin{aligned} \varphi^{(n+1)}(t) &= R_n^{(n+1)}(t) - \frac{R_n(x)}{\omega(x)}\omega^{(n+1)}(t) \\ &= f^{(n+1)}(t) - \frac{R_n(x)}{\omega(x)}(n+1)!, \end{aligned}$$

从而由 $\varphi^{(n+1)}(\xi) = 0$ 立刻得到 (4.4) 式.

由误差公式 (4.4) 可知, 如果 $f^{(n+1)}(x)$ 在 $[a,b]$ 上有界, 即存在常数 M, 使得 $|f^{(n+1)}(x)| \leqslant M$, 则必有

$$|R_n(x)| \leqslant \frac{M}{(n+1)!}|x-x_0|\cdot|x-x_1|\cdots|x-x_n|. \tag{4.6}$$

例 4.1 已知 $\omega(x)=\prod\limits_{i=0}^{n}(x-x_i)$, 求证

$$\omega'(x_k)=\prod_{i=0,i\neq k}^{n}(x_k-x_i),\quad k=1,2,\cdots,n.$$

证 因为

$$\omega(x)=\prod_{i=0}^{n}(x-x_i)=(x-x_k)\prod_{i=0,i\neq k}^{n}(x-x_i),\quad k=1,2,\cdots,n.$$

求导数得

$$\omega'(x)=\prod_{i=0,i\neq k}^{n}(x-x_i)+(x-x_k)\frac{\mathrm{d}}{\mathrm{d}x}\Big[\prod_{i=0,i\neq k}^{n}(x-x_i)\Big].$$

由此即得

$$\omega'(x_k)=\prod_{i=0,i\neq k}^{n}(x_k-x_i),\quad k=1,2,\cdots,n.$$

证毕. □

4.1.3 拉格朗日插值及其通用程序

前面已经讨论过, 满足插值条件 (4.2) 的插值多项式 (4.1) 是唯一存在的, 它的系数可以通过求解线性方程组 (4.3) 得到. 但由于求解线性方程组的计算量较大, 且当 n 较大时, 方程组 (4.3) 是一个病态方程组, 求解不可靠. 我们可以通过"基函数法"得到拉格朗日插值多项式, 从而不必解线性方程组, 避免了范德蒙德矩阵的病态现象.

1. 线性插值

设已知 x_0, x_1 及 $y_0=f(x_0)$, $y_1=f(x_1)$, $L_1(x)$ 为不超过 1 次的多项式且满足 $L_1(x_0)=y_0$, $L_1(x_1)=y_1$. 几何上, $L_1(x)$ 为经过 (x_0,y_0), (x_1,y_1) 两点的直线, 从而得到

$$L_1(x)=y_0+\frac{y_1-y_0}{x_1-x_0}(x-x_0). \tag{4.7}$$

为了推广到高阶插值问题, 我们将 (4.7) 式变形为对称形式

$$L_1(x)=l_0(x)y_0+l_1(x)y_1, \tag{4.8}$$

其中

$$l_0(x)=\frac{x-x_1}{x_0-x_1},\quad l_1(x)=\frac{x-x_0}{x_1-x_0}$$

均为 1 次多项式, 且满足

$$\begin{aligned}&l_0(x_0)=1,\quad l_0(x_1)=0,\\&l_1(x_0)=0,\quad l_1(x_1)=1.\end{aligned}$$

上面的关系式可以统一写成

$$l_i(x_j)=\begin{cases}1, & i=j,\\0, & i\neq j,\end{cases}\qquad i,\ j=0,1. \tag{4.9}$$

2. **抛物插值**

设已知 x_0, x_1, x_2 及 $y_i=f(x_i)$, $i=0,1,2$, $L_2(x)$ 为不超过 2 次的多项式, 且满足插值条件 $L_2(x_i)=y_i\,(i=0,1,2)$. 由 $L_1(x)$ 的表达式 (4.8) 猜测到, $L_2(x)$ 应有下述表达式

$$L_2(x)=l_0(x)y_0+l_1(x)y_1+l_2(x)y_2, \tag{4.10}$$

其中, $l_0(x)$, $l_1(x)$, $l_2(x)$ 均为 2 次多项式且满足 (4.9) $(i,j=0,1,2)$. 可以验证, 由 (4.10) 给出的 $L_2(x)$ 满足插值条件. 事实上,

$$L_2(x_0)=l_0(x_0)y_0+l_1(x_0)y_1+l_2(x_0)y_2=1\times y_0+0\times y_1+0\times y_2=y_0.$$

同理, $L_2(x_1)=y_1$, $L_2(x_2)=y_2$. 因此, 猜测式 (4.10) 是正确的, 于是问题转化为求 $l_i(x)\,(i=0,1,2)$.

因 $l_0(x)$ 是 2 次多项式且 $l_0(x_1)=l_0(x_2)=0$, 故有

$$l_0(x)=c(x-x_1)(x-x_2).$$

再由 $l_0(x_0)=c(x_0-x_1)(x_0-x_2)=1$, 得

$$c=\frac{1}{(x_0-x_1)(x_0-x_2)},$$

从而

$$l_0(x)=\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}. \tag{4.11}$$

同理,

$$l_1(x)=\frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)},\quad l_2(x)=\frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}. \tag{4.12}$$

3. n 阶拉格朗日插值

设已知 $x_0, x_1, \cdots, x_n$ 及 $y_i = f(x_i)\,(i = 0, 1, \cdots, n)$, $L_n(x)$ 为不超过 n 次的多项式, 且满足插值条件 $L_n(x_i) = y_i\,(i = 0, 1, \cdots, n)$. 由对 $L_2(x)$ 的构造经验, 可设

$$L_n(x) = \sum_{i=0}^{n} l_i(x)y_i = l_0(x)y_0 + l_1(x)y_1 + \cdots + l_n(x)y_n,$$

其中, $l_i(x)\,(i = 0, 1, \cdots, n)$ 均为 n 次多项式且满足 (4.9) $(i, j = 0, 1, \cdots, n)$. 不难验证, 这样构造出的 $L_n(x)$ 满足插值条件. 因此问题归结为求 $l_i(x)\,(i = 0, 1, \cdots, n)$ 的表达式. 因 $x_j(j \neq i)$ 是 n 次多项式 $l_i(x)$ 的 n 个根, 故可设

$$l_i(x) = c(x - x_0)\cdots(x - x_{i-1})(x - x_{i+1})\cdots(x - x_n) = c\prod_{j=0, j\neq i}^{n}(x - x_j).$$

再由

$$l_i(x_i) = c\prod_{j=0, j\neq i}^{n}(x_i - x_j) = 1, \quad i = 0, 1, \cdots, n,$$

得

$$c = \frac{1}{\prod\limits_{j=0, j\neq i}^{n}(x_i - x_j)}.$$

故有

$$L_n(x) = \sum_{i=0}^{n} l_i(x)y_i, \quad l_i(x) = \prod_{j=0, j\neq i}^{n}\frac{x - x_j}{x_i - x_j}. \tag{4.13}$$

公式 (4.13) 称为 n 阶拉格朗日插值公式, 其中 $l_i(x)\,(i = 0, 1, \cdots, n)$ 称为 n 阶拉格朗日插值的基函数.

例 4.2 已知函数表 $\sin\frac{\pi}{6} = 0.5000$, $\sin\frac{\pi}{4} = 0.7071$, $\sin\frac{\pi}{3} = 0.8660$, 分别由线性插值和抛物插值求 $\sin\frac{2\pi}{9}$ 的近似值, 并估计其精度.

解 (1) 线性插值只需要两个节点, 根据余项公式选取前两个节点.

$$\sin\frac{2\pi}{9} \approx L_1\left(\frac{2\pi}{9}\right) = \frac{\frac{2\pi}{9} - \frac{\pi}{4}}{\frac{\pi}{6} - \frac{\pi}{4}} \times 0.5000 + \frac{\frac{2\pi}{9} - \frac{\pi}{6}}{\frac{\pi}{4} - \frac{\pi}{6}} \times 0.7071$$

$$= \frac{1}{3} \times 0.5000 + \frac{2}{3} \times 0.7071 = 0.6381.$$

截断误差为

$$\left|R_1\left(\frac{2\pi}{9}\right)\right| = \left|\frac{(\sin x)''}{2!}\left(\frac{2\pi}{9} - \frac{\pi}{6}\right)\left(\frac{2\pi}{9} - \frac{\pi}{4}\right)\right| \leqslant \frac{1}{2} \times \frac{\pi}{18} \times \frac{\pi}{36} = 7.615 \times 10^{-3},$$

得 $\varepsilon = 7.615 \times 10^{-3} < 0.5 \times 10^{-1}$, 因此计算结果至少有 1 位有效数字.

(2) 由 (4.10), (4.11) 和 (4.12) 得

$$\begin{aligned}\sin\frac{2\pi}{9} &\approx L_2\left(\frac{2\pi}{9}\right)\\ &= \frac{\left(\frac{2\pi}{9}-\frac{\pi}{4}\right)\left(\frac{2\pi}{9}-\frac{\pi}{3}\right)}{\left(\frac{\pi}{6}-\frac{\pi}{4}\right)\left(\frac{\pi}{6}-\frac{\pi}{3}\right)}\times 0.5000+\frac{\left(\frac{2\pi}{9}-\frac{\pi}{6}\right)\left(\frac{2\pi}{9}-\frac{\pi}{3}\right)}{\left(\frac{\pi}{4}-\frac{\pi}{6}\right)\left(\frac{\pi}{4}-\frac{\pi}{3}\right)}\times 0.7071\\ &\quad+\frac{\left(\frac{2\pi}{9}-\frac{\pi}{6}\right)\left(\frac{2\pi}{9}-\frac{\pi}{4}\right)}{\left(\frac{\pi}{3}-\frac{\pi}{6}\right)\left(\frac{\pi}{3}-\frac{\pi}{4}\right)}\times 0.8660\\ &= \frac{2}{9}\times 0.5+\frac{8}{9}\times 0.7071-\frac{1}{9}\times 0.866 = 0.6434.\end{aligned}$$

截断误差为

$$\begin{aligned}\left|R_2\left(\frac{2\pi}{9}\right)\right| &= \left|\frac{(\sin x)'''}{3!}\left(\frac{2\pi}{9}-\frac{\pi}{6}\right)\left(\frac{2\pi}{9}-\frac{\pi}{4}\right)\left(\frac{2\pi}{9}-\frac{\pi}{3}\right)\right|\\ &\leqslant \frac{1}{6}\times\frac{\pi}{18}\times\frac{\pi}{36}\times\frac{\pi}{36} = 8.861\times 10^{-4},\end{aligned}$$

得 $\varepsilon = 8.861 \times 10^{-4} < 0.5 \times 10^{-2}$, 因此计算结果至少有 2 位有效数字.

4. 拉格朗日插值公式的通用程序

根据拉格朗日插值公式 (4.13), 编写下列程序, 可对于给定的数据求得插值点的插值结果. 注意, 该程序并不能输出插值多项式的表达式.

- **拉格朗日插值 MATLAB 程序**

```
%malagr.m
function yy=malagr(x,y,xx)
%用途：拉格朗日插值法求解
%格式：yy=malagr(x,y,xx)，x是节点向量，y是节点对应的函
%数值向量，xx是插值点(可以是多个)，yy返回插值结果
m=length(x); n=length(y);
if m~=n,
   error('向量x与y的长度必须一致');
end
```

```
s=0;
for i=1:n
    t=ones(1,length(xx));
    for j=1:n
        if j~=i
            t=t.*(xx-x(j))/(x(i)-x(j));
        end
    end
    s=s+t*y(i);
end
yy=s;
```

例 4.3 *用上面的程序 malagr.m 求解例 4.2.*

解 在 MATLAB 命令窗口执行下列命令：

```
>> x=pi*[1/6 1/4];
>> y=[0.5 0.7071]; xx=2*pi/9;
>> yy1=malagr(x,y,xx)
yy1 =
    0.6381
>> x=pi*[1/6 1/4 1/3];
>> y=[0.5 0.7071 0.8660]; xx=2*pi/9;
>> yy2=malagr(x,y,xx)
yy2 =
    0.6434
```

4.1.4 Hermite 插值

拉格朗日插值仅考虑节点的函数值约束，而一些插值问题还需要在某些节点具有插值函数与被插值函数的导数值的一致性. 具有节点的导数值约束的插值称为 Hermite 插值. 下面我们采取与拉格朗日插值完全平行的过程讨论一种特殊的 3 阶 Hermite 插值多项式的构造及其余项，它与样条插值有密切联系.

已知 x_0, x_1, $y_0 = f(x_0)$, $y_1 = f(x_1)$ 及 $y_0' = f'(x_0)$, $y_1' = f'(x_1)$, 求不超过 3 次的多项式 $H_3(x)$ 使满足

$$H_3(x_0) = y_0,\quad H_3(x_1) = y_1,\quad H_3'(x_0) = y_0',\quad H_3'(x_1) = y_1'. \tag{4.14}$$

首先，当 $x_0 \neq x_1$, 不难证明, $H_3(x)$ 存在唯一.

其次, 用基函数法导出 $H_3(x)$ 的计算公式. 记 $h = x_1 - x_0$, 引入变量代换

$$\bar{x} = \frac{x - x_0}{h},$$

并令 $\bar{f}(\bar{x}) = f(x)$, 则 $\bar{f}(0) = y_0$, $\bar{f}(1) = y_1$ 及 $\bar{f}'(0) = y_0'$, $\bar{f}'(1) = y_1'$. 参照 n 阶拉格朗日插值多项式的"基函数法", 令

$$H_3(x) = \alpha_0(\bar{x})y_0 + \alpha_1(\bar{x})y_1 + h\beta_0(\bar{x})y_0' + h\beta_1(\bar{x})y_1', \tag{4.15}$$

其中 $\alpha_0(\bar{x})$, $\alpha_1(\bar{x})$, $h\beta_0(\bar{x})$, $\beta_1(\bar{x})$ 均为 3 次多项式, 且满足

$$\begin{array}{llll} \alpha_0(0) = 1, & \alpha_1(0) = 0, & \beta_0(0) = 0, & \beta_1(0) = 0, \\ \alpha_0(1) = 0, & \alpha_1(1) = 1, & \beta_0(1) = 0, & \beta_1(1) = 0, \\ \alpha_0'(0) = 0, & \alpha_1'(0) = 0, & \beta_0'(0) = 1, & \beta_1'(0) = 0, \\ \alpha_0'(1) = 0, & \alpha_1'(1) = 0, & \beta_0'(1) = 0, & \beta_1'(1) = 1. \end{array}$$

由 $\alpha_0(\bar{x})$ 的第 2 和第 4 个约束条件, 可设 $\alpha_0(\bar{x}) = (a\bar{x} + b)(\bar{x} - 1)^2$, 再利用第 1 和第 3 个约束条件可得 $a = 2$, $b = 1$. 这样, $\alpha_0(\bar{x}) = (2\bar{x} + 1)(\bar{x} - 1)^2$. 类似可求出 $\alpha_1(\bar{x})$, $\beta_0(\bar{x})$, $\beta_1(\bar{x})$ 的表达式. 我们有

$$\begin{aligned} &\alpha_0(\bar{x}) = 2\bar{x}^3 - 3\bar{x}^2 + 1, \quad \alpha_1(\bar{x}) = -2\bar{x}^3 + 3\bar{x}^2, \\ &\beta_0(\bar{x}) = \bar{x}^3 - 2\bar{x}^2 + \bar{x}, \quad \beta_1(\bar{x}) = \bar{x}^3 - \bar{x}^2. \end{aligned} \tag{4.16}$$

故所求的 3 阶 Hermite 插值多项式 $H_3(x)$ 为

$$\begin{aligned} H_3(x) = &\alpha_0\left(\frac{x - x_0}{h}\right)y_0 + \alpha_1\left(\frac{x - x_0}{h}\right)y_1 \\ &+ h\beta_0\left(\frac{x - x_0}{h}\right)y_0' + h\beta_1\left(\frac{x - x_0}{h}\right)y_1'. \end{aligned} \tag{4.17}$$

最后, 导出 $H_3(x)$ 的余项 $R_3(x) = f(x) - H_3(x)$. 构造辅助函数:

$$\varphi(t) = R_3(t) - \frac{R_3(x)}{\pi(x)}\pi(t), \quad \pi(t) = (t - x_0)^2(t - x_1)^2.$$

类似于拉格朗日插值余项的推导过程, 并注意到 $\varphi'(x_0) = \varphi'(x_1) = 0$, 可导出

$$R_3(x) = f(x) - H_3(x) = \frac{f^{(4)}(\xi)}{4!}(x - x_0)^2(x - x_1)^2, \tag{4.18}$$

其中, x_0, x_1, $x \in [a, b]$, $f(x)$ 在 $[a, b]$ 上有 4 阶连续导数, ξ 介于 x 及 x_0, x_1 之间.

例 4.4 设 $f(x) = \ln x$, 给定 $f(1) = 0$, $f(2) = 0.69315$, $f'(1) = 1$, $f'(2) = 0.5$. 用 3 次 Hermite 插值多项式 $H_3(x)$ 来计算 $f(1.5)$ 的近似值.

解 这里 $x_0=1$, $x_1=2$, $h=x_1-x_0=1$. 则由 (4.16) 和 (4.17) 得

$$\begin{aligned}H_3(1.5)=&\alpha_0\Big(\frac{1.5-1}{1}\Big)\times 0+\alpha_1\Big(\frac{1.5-1}{1}\Big)\times 0.69315\\&+1\times\beta_0\Big(\frac{1.5-1}{1}\Big)\times 1+1\times\beta_1\Big(\frac{1.5-1}{1}\Big)\times 0.5\\=&0.69315\times\alpha_1(0.5)+\beta_0(0.5)+0.5\times\beta_1(0.5).\end{aligned}$$

注意到

$$\alpha_1(0.5)=-2\times 0.5^3+3\times 0.5^2=0.5,$$

$$\beta_0(0.5)=0.5^3-2\times 0.5^2+0.5=0.125,$$

$$\beta_1(0.5)=0.5^3-0.5^2=-0.125,$$

故

$$f(1.5)\approx H_3(1.5)=0.69315\times 0.5+0.125-0.5\times 0.125=0.409075.$$

4.2 牛顿插值法

由于拉格朗日插值公式计算缺少递推关系, 每次新增加节点需要重新计算, 高次插值无法利用低次插值的结果. 通过引进差商的概念, 可以给出一种在增加节点时可对拉格朗日插值多项式进行递推计算的方法. 该方法称为牛顿插值法.

4.2.1 差商及其性质

我们首先给出差商的定义.

定义 4.1 *设已知 $x_0, x_1, \cdots, x_n$, 称*

$$f[x_0,x_k]=\frac{f(x_k)-f(x_0)}{x_k-x_0},\quad k=1,2,\cdots,n$$

为 $f(x)$ 关于节点 x_0, x_k 的 1 阶差商. 称

$$f[x_0,x_1,x_k]=\frac{f[x_0,x_k]-f[x_0,x_1]}{x_k-x_1},\quad k=2,\cdots,n$$

为 $f(x)$ 关于节点 x_0, x_1, x_k 的 2 阶差商. 一般地, 若定义了 $k-1$ 阶差商, 则称

$$f[x_0,x_1,\cdots,x_k]=\frac{f[x_0,\cdots,x_{k-2},x_k]-f[x_0,\cdots,x_{k-2},x_{k-1}]}{x_k-x_{k-1}},\quad k\leqslant n$$

为 $f(x)$ 关于节点 $x_0, x_1, \cdots, x_k$ 的 k 阶差商.

例 4.5 设已知 $f(0)=1$, $f(-1)=5$, $f(2)=-1$, 分别求 $f[0,-1,2]$ 和 $f[-1,2,0]$.

解 因

$$f[0,-1]=\frac{f(-1)-f(0)}{-1-0}=-4,\ f[0,2]=\frac{f(2)-f(0)}{2-0}=-1,$$

故

$$f[0,-1,2]=\frac{f[0,2]-f[0,-1]}{2-(-1)}=\frac{(-1)-(-4)}{3}=1.$$

又

$$f[-1,2]=\frac{f(2)-f(-1)}{2-(-1)}=-2,\ \ f[-1,0]=\frac{f(0)-f(-1)}{0-(-1)}=-4,$$

所以

$$f[-1,2,0]=\frac{f[-1,0]-f[-1,2]}{0-2}=\frac{(-4)-(-2)}{-2}=1.$$

由本例可知, $f[0,-1,2]=f[-1,2,0]$, 这不是偶然的. 事实上, 差商具有下列性质:

定理 4.1(差商的性质) (1) 成立

$$f[x_0,x_1,\cdots,x_k]=\sum_{i=0}^{k}\frac{f(x_i)}{\prod\limits_{j=0,j\neq i}^{k}(x_i-x_j)}=\sum_{i=0}^{k}\frac{f(x_i)}{\omega'(x_i)},\tag{4.19}$$

其中 $\omega(x)=\prod\limits_{j=0}^{n}(x-x_j)$.

(2) 差商与节点的排列次序无关.

证 性质 (2) 由 (4.19) 式的对称性立即可得. 我们用数学归纳法证明性质 (1). 当 $k=1$ 时,

$$\sum_{i=0}^{1}\frac{f(x_i)}{\prod\limits_{j=0,j\neq i}^{1}(x_i-x_j)}=\frac{f(x_0)}{x_0-x_1}+\frac{f(x_1)}{x_1-x_0}=\frac{f(x_1)-f(x_0)}{x_1-x_0}=f[x_0,x_1],$$

知 (4.19) 成立. 设对 $k-1$ 阶差商 (4.19) 式成立. 对 $i=0,\cdots,k-2$, 记 $y_i=x_i$ 及

$y_{k-1}=x_k$, 得

$$f[x_0,\cdots,x_{k-1},x_k]=\frac{f[x_0,\cdots,x_{k-2},x_k]-f[x_0,\cdots,x_{k-2},x_{k-1}]}{x_k-x_{k-1}}$$

$$=\frac{1}{x_k-x_{k-1}}\left\{\sum_{i=0}^{k-1}\frac{f(y_i)}{\prod\limits_{j=0,j\neq i}^{k-1}(y_i-y_j)}-\sum_{i=0}^{k-1}\frac{f(x_i)}{\prod\limits_{j=0,j\neq i}^{k-1}(x_i-x_j)}\right\}.$$

$$=\frac{1}{x_k-x_{k-1}}\left\{\sum_{i=0}^{k-2}f(x_i)\left[\frac{1}{\left(\prod\limits_{j=0,j\neq i}^{k-1}(x_i-x_j)\right)(x_i-x_k)}-\frac{1}{\prod\limits_{j=0,j\neq i}^{k-1}(x_i-x_j)}\right]\right.$$

$$\left.+\frac{f(x_k)}{\prod\limits_{j=0,j\neq i}^{k-2}(x_k-x_j)}-\frac{f(x_{k-1})}{\prod\limits_{j=0,j\neq i}^{k-2}(x_{k-1}-x_j)}\right\}$$

$$=\sum_{i=0}^{k-2}\frac{f(x_i)}{\prod\limits_{j=0,j\neq i}^{k}(x_i-x_j)}+\frac{f(x_{k-1})}{\prod\limits_{j=0,j\neq k-1}^{k}(x_{k-1}-x_j)}+\frac{f(x_k)}{\prod\limits_{j=0,j\neq k}^{k}(x_k-x_j)}$$

$$=\sum_{i=0}^{k}\frac{f(x_i)}{\prod\limits_{j=0,j\neq i}^{k}(x_i-x_j)}=\sum_{i=0}^{k}\frac{f(x_i)}{\omega'(x_i)}.$$

由数学归纳法知 (4.19) 式成立. □

4.2.2 牛顿插值公式

设已知 $x_0,x_1,\cdots,x_n$ 及 $y_i=f(x_i)\,(i=0,1,\cdots,n)$, 由差商的定义, 当 $x\neq x_i\,(i=0,1,\cdots,n)$ 时, 由

$$f[x_0,x]=\frac{f(x)-f(x_0)}{x-x_0}\quad\Rightarrow\quad f(x)=f(x_0)+f[x_0,x](x-x_0),$$

$$f[x_0,x_1,x]=\frac{f[x_0,x]-f[x_0,x_1]}{x-x_1}$$

$$\Rightarrow\quad f[x_0,x]=f[x_0,x_1]+f[x_0,x_1,x](x-x_1),$$

从而

$$f(x)=f(x_0)+f[x_0,x_1](x-x_0)+f[x_0,x_1,x](x-x_0)(x-x_1).$$

依此类推, 得到

$$\begin{aligned}f(x)=&f(x_0)+f[x_0,x_1](x-x_0)+f[x_0,x_1,x_2](x-x_0)(x-x_1)\\&+\cdots+f[x_0,x_1,\cdots,x_n](x-x_0)(x-x_1)\cdots(x-x_{n-1})\\&+f[x_0,\cdots,x_n,x](x-x_0)\cdots(x-x_{n-1})(x-x_n)\\=&N_n(x)+R_n(x),\end{aligned}$$

其中

$$\begin{aligned}N_n(x)=&f(x_0)+f[x_0,x_1](x-x_0)+f[x_0,x_1,x_2](x-x_0)(x-x_1)\\&+\cdots+f[x_0,\cdots,x_n](x-x_0)(x-x_1)\cdots(x-x_{n-1})\end{aligned} \tag{4.20}$$

及

$$R_n(x) = f[x_0,\cdots,x_n,x]\omega(x), \quad \omega(x)=\prod_{j=0}^{n}(x-x_j). \tag{4.21}$$

由于 $N_n(x)$ 为不超过 n 次的多项式, 且满足

$$N_n(x_i)=f(x_i)-R_n(x_i)=f(x_i)=y_i,\quad i=0,1,\cdots,n,$$

故由插值多项式的唯一性知, $N_n(x)=L_n(x)$ 恰为 $f(x)$ 关于节点 $x_0,x_1,\cdots,x_n$ 的拉格朗日插值多项式. 再由

$$R_n(x)=f(x)-N_n(x)=f(x)-L_n(x)=\frac{f^{(n+1)}(\xi)}{(n+1)!}\omega(x),$$

结合 (4.21) 即得

$$f[x_0,\cdots,x_n,x]=\frac{f^{(n+1)}(\xi)}{(n+1)!},\ \xi \text{ 介于 } x_0,\cdots,x_n \text{ 及 } x \text{ 之间}.$$

从而有

$$f[x_0,\cdots,x_n]=\frac{f^{(n)}(\xi)}{n!},\ \xi \text{ 介于 } x_0,\cdots,x_n \text{ 之间}. \tag{4.22}$$

由 (4.20) 给出的 $N_n(x)$ 称为 $f(x)$ 关于节点 $x_0,x_1,\cdots,x_n$ 的牛顿插值多项式. 这种方法在增加节点时可方便地进行递推计算.

例 4.6 *用牛顿插值求解例 4.2. 若进一步利用 $\sin\frac{\pi}{2}=1$ 应如何计算?*

解 $f(x)$ 关于节点 $\frac{\pi}{6}, \frac{\pi}{4}, \frac{\pi}{3}$ 的各阶差商计算结果如下:

x_k	$f(x_k)$	$f[x_0,x_k]$	$f[x_0,x_1,x_k]$
$\pi/6$	0.5000		
$\pi/4$	0.7071	0.7911	
$\pi/3$	0.8660	0.6990	−0.3518

从而由牛顿插值公式 (4.20) 得

线性插值:

$$\sin\frac{2\pi}{9} \approx N_1\left(\frac{2\pi}{9}\right) = 0.5000 + 0.7911 \times \left(\frac{2\pi}{9} - \frac{\pi}{6}\right) = 0.6381.$$

抛物插值:

$$\sin\frac{2\pi}{9} \approx N_2\left(\frac{2\pi}{9}\right) = N_1\left(\frac{2\pi}{9}\right) - 0.3518 \times \left(\frac{2\pi}{9} - \frac{\pi}{6}\right) \times \left(\frac{2\pi}{9} - \frac{\pi}{4}\right)$$

$$= 0.6381 + 0.3518 \times \frac{\pi}{18} \times \frac{\pi}{36} = 0.6434.$$

进一步利用 $\sin\frac{\pi}{2}$ 得 3 阶差商如下:

x_k	$f(x_k)$	$f[x_0, x_k]$	$f[x_0, x_1, x_k]$	$f[x_0, x_1, x_2, x_k]$
$\pi/6$	0.5000			
$\pi/4$	0.7071	0.7911		
$\pi/3$	0.8660	0.6990	−0.3518	
$\pi/2$	1.0000	0.4775	−0.3993	−0.09072

可得

$$\sin\frac{2\pi}{9} \approx N_3\left(\frac{2\pi}{9}\right)$$

$$= N_2\left(\frac{2\pi}{9}\right) - 0.09072 \times \left(\frac{2\pi}{9} - \frac{\pi}{6}\right) \times \left(\frac{2\pi}{9} - \frac{\pi}{4}\right) \times \left(\frac{2\pi}{9} - \frac{\pi}{3}\right)$$

$$= 0.6434 - 0.09072 \times \frac{\pi}{18} \times \frac{\pi}{36} \times \frac{\pi}{9} = 0.6429.$$

对照例 4.2 的运算过程可见, 使用牛顿插值各阶插值之间有递推关系, 当增加节点时计算要方便得多.

4.3 样条插值法

4.3.1 高阶插值的 Runge 现象

从拉格朗日插值余项公式的分母部分可以发现, 节点数的增加对提高插值精度是有利的, 但这只是问题的一个方面. 以下讨论的著名例子揭示了问题的另一方面, 即并非插值节点越多精度越高.

例 4.7 设

$$f(x) = \frac{1}{1+x^2},$$

分别讨论将区间 $[-5,5]$ 5 等分和 10 等分后, 拉格朗日插值的效果.

解 根据拉格朗日插值法通用程序 malagr.m, 编写下面的 MATLAB 程序:

```
%marunge.m
function marunge()
xx=-5:0.05:5;
y=1./(1+xx.^2);
x1=-5:2:5;
y1=1./(1+x1.^2);
x2=-5:1:5;
y2=1./(1+x2.^2);
yy1=malagr(x1,y1,xx);
yy2=malagr(x2,y2,xx);
plot(xx,yy1,'-.');
hold on
plot(xx,yy2,'r-.');
plot(xx,y);
text(1.5,0.5,'L5');
text(4.0,1.5,'L10');
```

在 MATLAB 命令窗口执行

```
>> marunge
```

得到 $L_5(x)$ 和 $L_{10}(x)$ 的图像, 如图 4.1.

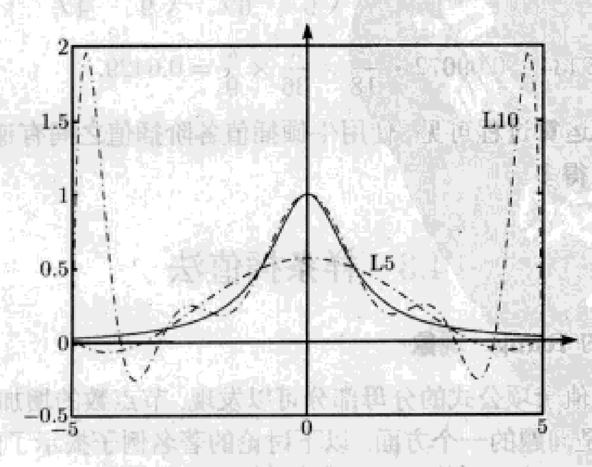


图 4.1 高阶插值的 Runge 现象

分析插值结果可以发现, 在 $[-5,-4]\cap[4,5]$ 部分, $L_{10}(x)$ 比 $L_5(x)$ 效果更差, 这种高阶插值的振荡现象称为 Runge 现象. 从拉格朗日插值余项公式 (4.4) 不难找出

Runge 现象发生的原因. 事实上, 由于

$$f(x)=\frac{1}{1+x^2}=\frac{1}{2i}\Big(\frac{1}{x-i}-\frac{1}{x+i}\Big),$$

可得

$$f^{(n+1)}(x)=\frac{(-1)^{n+1}}{2i}(n+1)!\Big[\frac{1}{(x-i)^{n+2}}-\frac{1}{(x+i)^{n+2}}\Big].$$

从而

$$\begin{aligned}f(x)-L_n(x)&=\frac{(-1)^{n+1}}{2i}(n+1)!\Big[\frac{1}{(\xi-i)^{n+2}}-\frac{1}{(\xi+i)^{n+2}}\Big]\omega(x)\\&=\frac{(-1)^{n+1}\omega(x)}{(\xi^2+1)^{n/2+1}}\sin(n+2)\theta,\end{aligned}$$

其中, $\theta=\arctan\frac{1}{\xi}$. 当 $n\to\infty$ 时, 无法保证余项的收敛性.

4.3.2 分段插值

避免高阶插值 Runge 现象的基本方法是使用分段函数进行分段插值. 我们在此介绍分段线性插值 $I_1(x)$ 和分段 3 阶 Hermite 插值 $I_3(x)$.

1. 分段线性插值

设已知 $x_0<x_1<\cdots<x_n$ 及 $y_i=f(x_i)\,(i=0,1,\cdots,n)$, $I_1(x)$ 为区间 $[x_{i-1},x_i]$ 上不超过 1 次的多项式, 且满足 $I_1(x_{i-1})=y_{i-1}$, $I_1(x_i)=y_i\,(i=0,1,\cdots,n)$. 由线性插值公式得

$$I_1(x)=\frac{x-x_i}{x_{i-1}-x_i}y_{i-1}+\frac{x-x_{i-1}}{x_i-x_{i-1}}y_i,\quad x_{i-1}\leqslant x\leqslant x_i. \tag{4.23}$$

可以证明, 分段线性插值是收敛的. 事实上, 分段线性插值的余项为

$$R_1(x)=f(x)-I_1(x)=\frac{f''(\xi)}{2}(x-x_{i-1})(x-x_i),\quad x_{i-1}\leqslant x\leqslant x_i. \tag{4.24}$$

由于

$$\max_{x_{i-1}\leqslant x\leqslant x_i}|(x-x_{i-1})(x-x_i)|=\frac{(x_i-x_{i-1})^2}{4},$$

故由 (4.24) 得误差上界

$$|R_1(x)|=|f(x)-I_1(x)|\leqslant\frac{h^2}{8}M_2, \tag{4.25}$$

其中

$$h=\max_{1\leqslant i\leqslant n}(x_i-x_{i-1}),\quad M_2=\max_{x_0\leqslant x\leqslant x_n}|f''(x)|.$$

由 (4.25) 立知, 当 $h \to 0$ 时, $I_1(x) \to f(x)$.

分段线性插值简单, 易于应用. 同时由 (4.25) 可知, 可通过选取适当的步长 h 控制精度. 但它不具有光滑性, 使用 Hermite 插值原理可得到具有光滑性的分段插值.

2. 分段 3 阶 Hermite 插值

设已知 $x_0 < x_1 < \cdots < x_n$, $y_i = f(x_i)$ 及 $y_i' = f'(x_i)\,(i = 0, 1, \cdots, n)$. $I_3(x)$ 为区间 $[x_{i-1}, x_i]$ 上不超过 3 次的多项式, 且满足

$$\begin{cases} I_3(x_{i-1}) = y_{i-1}, \quad I_3(x_i) = y_i, \\ I_3'(x_{i-1}) = y_{i-1}', \quad I_3'(x_i) = y_i', \end{cases} \quad i = 1, \cdots, n.$$

由 3 阶 Hermite 插值公式 (4.17) 得

$$\begin{aligned} I_3(x) = {} & \alpha_0\left(\frac{x - x_{i-1}}{h_i}\right) y_i + \alpha_1\left(\frac{x - x_{i-1}}{h_i}\right) y_{i-1} \\ & + h_i\beta_0\left(\frac{x - x_{i-1}}{h_i}\right) y_{i-1}' + \beta_1\left(\frac{x - x_{i-1}}{h_i}\right) y_i', \\ & x_{i-1} \leqslant x \leqslant x_i, \; i = 1, \cdots, n, \end{aligned} \tag{4.26}$$

其中, $h_i = x_i - x_{i-1}$, 基函数 $\alpha_0(x)$, $\alpha_1(x)$, $\beta_0(x)$, $\beta_1(x)$ 的表达式见 (4.16).

再来看看分段 3 阶 Hermite 插值的截断误差. 由 (4.18) 可得

$$\begin{aligned} R_3(x) = f(x) - I_3(x) = \frac{f^{(4)}(\xi)}{4!}(x - x_{i-1})^2(x - x_i)^2, \\ x_{i-1} \leqslant x \leqslant x_i, \; i = 1, \cdots, n. \end{aligned} \tag{4.27}$$

由此得误差估计式

$$|R_3(x)| = |f(x) - I_3(x)| \leqslant \frac{h^2}{384} M_4, \tag{4.28}$$

其中

$$h = \max_{1 \leqslant i \leqslant n}(x_i - x_{i-1}), \quad M_4 = \max_{x_0 \leqslant x \leqslant x_n} |f^{(4)}(x)|.$$

由 (4.28) 立知, 当 $h \to 0$ 时, $I_3(x) \to f(x)$, 即分段 3 阶 Hermite 插值是收敛的.

分段 3 阶 Hermite 插值多项式显然有 1 阶连续导数, 但在实际应用中一般不知道, 我们可以利用这一自由度得到光滑性更好的、在实际工程计算中应用最广泛的三阶样条插值.

4.3.3 三阶样条插值及其通用程序

1. 算法推导

设已知 $x_0 < x_1 < \cdots < x_n$ 及 $y_i = f(x_i)\,(i=0,1,\cdots,n)$, 插值函数 $S(x)$ 在每个小区间 $[x_{i-1},x_i]$ 上是不超过 3 次的多项式且具有 2 阶连续导数, 则称 $S(x)$ 为三阶样条插值. 具体地说, 三阶样条插值是满足下列条件的分段三次多项式:

(1) 插值条件:

$$S(x_i) = y_i \quad (i=0,1,\cdots,n);$$

(2) 连接条件:

$$S(x_i-0)=S(x_i+0), \quad S'(x_i-0)=S'(x_i+0),$$
$$S''(x_i-0)=S''(x_i+0), \quad i=1,\cdots,n-1.$$

我们来分析一下三阶样条插值的存在性. 这里, $S(x)$ 是 n 个不超过 3 次的多项式, 共含 $4n$ 个待定参数. 插值条件给出了 $n+1$ 个约束, 连接条件给出了 $3(n-1)$ 个约束. 故插值条件和连接条件一共给出了 $4n-2$ 个约束. 与待定参数相比还少 2 个约束, 为此可按实际需要, 人为地添加 2 个边界条件. 常用的边界条件有下列 4 类:

(1) **一阶导数**: $S'(x_0)=y_0'$, $S'(x_n)=y_n'$;

(2) **二阶导数**: $S''(x_0)=y_0''$, $S''(x_n)=y_n''$;

(3) **周期样条**: $S'(x_0)=S'(x_n)$, $S''(x_0)=S''(x_n)$, 前提条件是 $S(x_0)=S(x_n)$, 当被插函数是周期函数或封闭曲线时, 宜用周期条件;

(4) **非扭结**: 第一段和第二段多项式 3 次项系数相同, 最后一段和倒数第二段多项式 3 次项系数相同.

下面我们利用分段 3 阶 Hermite 插值给出三阶样条插值的一种算法.

设 $S'(x_i)=m_i\,(i=0,1,\cdots,n)$, 由分段 3 阶 Hermite 插值式 (4.26) 得

$$\begin{aligned} S(x) = {} & \alpha_0\left(\frac{x-x_{i-1}}{h_i}\right)y_{i-1} + \alpha_1\left(\frac{x-x_{i-1}}{h_i}\right)y_i \\ & + h_i\beta_0\left(\frac{x-x_{i-1}}{h_i}\right)m_{i-1} + h_i\beta_1\left(\frac{x-x_{i-1}}{h_i}\right)m_i, \qquad (4.29) \\ & x_{i-1} \leqslant x \leqslant x_i, \ i=1,\cdots,n, \end{aligned}$$

其中, $h_i = x_i - x_{i-1}$, 基函数 $\alpha_0(x)$, $\alpha_1(x)$, $\beta_0(x)$, $\beta_1(x)$ 的表达式由 (4.16) 定义.

由分段 3 阶 Hermite 插值的性质在写出 $S(x)$ 的表达式的过程中实际上已使插值条件和连接条件的连续性和一阶光滑性得到满足, 故余下的只需要根据 $n-1$

个二阶光滑性约束条件 $S''(x_i-0)=S''(x_i+0)$ 和 2 个边界条件来求 $n+1$ 个待定参数 $m_0, m_1, \cdots, m_n$. 直接计算得

$$\alpha_0''(x)=12x-6,\quad \alpha_1''(x)=-12x+6,$$

$$\beta_0''(x)=6x-4,\quad \beta_1''(x)=6x-2.$$

由此不难求得

$$\alpha_0''(1)=6,\quad \alpha_1''(1)=-6,\quad \beta_0''(1)=2,\quad \beta_1''(1)=4,$$

$$\alpha_0''(0)=-6,\quad \alpha_1''(0)=6,\quad \beta_0''(0)=-4,\quad \beta_1''(0)=-2.$$

利用 $[x_{i-1}, x_i]$ 上 $S(x)$ 的表达式求得

$$\begin{aligned}S''(x)=&\frac{y_{i-1}}{h_i^2}\alpha_0''\left(\frac{x-x_{i-1}}{h_i}\right)+\frac{y_i}{h_i^2}\alpha_1''\left(\frac{x-x_{i-1}}{h_i}\right)\\&+\frac{m_{i-1}}{h_i}\beta_0''\left(\frac{x-x_{i-1}}{h_i}\right)+\frac{m_i}{h_i}\beta_1''\left(\frac{x-x_{i-1}}{h_i}\right),\end{aligned}$$

故

$$\begin{aligned}S''(x_i-0)=&\frac{y_{i-1}}{h_i^2}\alpha_0''(1)+\frac{y_i}{h_i^2}\alpha_1''(1)+\frac{m_{i-1}}{h_i}\beta_0''(1)+\frac{m_i}{h_i}\beta_1''(1)\\=&\frac{2}{h_i}m_{i-1}+\frac{4}{h_i}m_i-\frac{6}{h_i^2}(y_i-y_{i-1}).\end{aligned}\tag{4.30}$$

同样, 利用 $[x_i, x_{i+1}]$ 上 $S(x)$ 的表达式求得

$$\begin{aligned}S''(x)=&\frac{y_i}{h_{i+1}^2}\alpha_0''\left(\frac{x-x_i}{h_{i+1}}\right)+\frac{y_{i+1}}{h_{i+1}^2}\alpha_1''\left(\frac{x-x_i}{h_{i+1}}\right)\\&+\frac{m_i}{h_{i+1}}\beta_0''\left(\frac{x-x_i}{h_{i+1}}\right)+\frac{m_{i+1}}{h_{i+1}}\beta_1''\left(\frac{x-x_i}{h_{i+1}}\right),\end{aligned}$$

故

$$\begin{aligned}S''(x_i+0)=&\frac{y_i}{h_{i+1}^2}\alpha_0''(0)+\frac{y_{i+1}}{h_{i+1}^2}\alpha_1''(0)+\frac{m_i}{h_{i+1}}\beta_0''(0)+\frac{m_{i+1}}{h_{i+1}}\beta_1''(0)\\=&-\frac{4}{h_{i+1}}m_i-\frac{2}{h_{i+1}}m_{i+1}+\frac{6}{h_{i+1}^2}(y_{i+1}-y_i).\end{aligned}\tag{4.31}$$

由 $S''(x_i-0)=S''(x_i+0)$ 得

$$\frac{2}{h_i}m_{i-1}+\frac{4}{h_i}m_i-\frac{6}{h_i^2}(y_i-y_{i-1})=-\frac{4}{h_{i+1}}m_i-\frac{2}{h_{i+1}}m_{i+1}+\frac{6}{h_{i+1}^2}(y_{i+1}-y_i),$$

即

$$\frac{h_{i+1}}{h_i+h_{i+1}}m_{i-1}+2m_i+\frac{h_i}{h_i+h_{i+1}}m_{i+1}$$
$$=\frac{3h_{i+1}}{h_i+h_{i+1}}\frac{y_i-y_{i-1}}{h_i}+\frac{3h_i}{h_i+h_{i+1}}\frac{y_{i+1}-y_i}{h_{i+1}}.$$

令

$$\lambda_i=\frac{h_{i+1}}{h_i+h_{i+1}},\quad \mu_i=1-\lambda_i,\quad \theta_i=3\lambda_i\frac{y_i-y_{i-1}}{h_i}+3\mu_i\frac{y_{i+1}-y_i}{h_{i+1}},\tag{4.32}$$

则有

$$\lambda_i m_{i-1}+2m_i+\mu_i m_{i+1}=\theta_i,\quad i=1,2,\cdots,n-1.\tag{4.33}$$

(1) 如果是第一类 (一阶导数) 边界条件, 则 $m_0=y_0'$, $m_n=y_n'$, 于是 (4.33) 可写成

$$\begin{pmatrix} 2 & \mu_1 & & & \\ \lambda_2 & 2 & \mu_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \lambda_{n-2} & 2 & \mu_{n-2} \\ & & & \lambda_{n-1} & 2 \end{pmatrix}\begin{pmatrix} m_1 \\ m_2 \\ \vdots \\ m_{n-2} \\ m_{n-1}\end{pmatrix}=\begin{pmatrix}\theta_1-\lambda_1 y_0' \\ \theta_2 \\ \vdots \\ \theta_{n-2} \\ \theta_{n-1}-\mu_{n-1}y_n'\end{pmatrix}.\tag{4.34}$$

(2) 如果是第二类 (二阶导数) 边界条件, 在 (4.31) 中取 $i=0$, 在 (4.30) 中取 $i=n$, 得

$$\begin{cases} -\dfrac{4}{h_1}m_0-\dfrac{2}{h_1}m_1+\dfrac{6}{h_1^2}(y_1-y_0)=y_0'', \\ \dfrac{2}{h_n}m_{n-1}+\dfrac{4}{h_n}m_n-\dfrac{6}{h_n^2}(y_n-y_{n-1})=y_n''. \end{cases}$$

整理得

$$\begin{cases} 2m_0+m_1=3\dfrac{y_1-y_0}{h_1}-\dfrac{h_1}{2}y_0'', \\ m_{n-1}+2m_n=3\dfrac{y_n-y_{n-1}}{h_n}+\dfrac{h_n}{2}y_n''. \end{cases}$$

令

$$\mu_0=1,\quad \lambda_n=1,\quad \theta_0=3\frac{y_1-y_0}{h_1},\quad \theta_n=3\frac{y_n-y_{n-1}}{h_n}.\tag{4.35}$$

这样, 结合 (4.32) 和 (4.33) 得

$$\begin{pmatrix} 2 & \mu_0 & & & \\ \lambda_1 & 2 & \mu_1 & & \\ & \ddots & \ddots & \ddots & \\ & & \lambda_{n-1} & 2 & \mu_{n-1} \\ & & & \lambda_n & 2 \end{pmatrix} \begin{pmatrix} m_0 \\ m_1 \\ \vdots \\ m_{n-1} \\ m_n \end{pmatrix} = \begin{pmatrix} \theta_0 - h_1 y_0''/2 \\ \theta_1 \\ \vdots \\ \theta_{n-1} \\ \theta_n + h_n y_n''/2 \end{pmatrix}. \quad (4.36)$$

方程组 (4.34) 和 (4.36) 都是三对角线性方程组, 可以用追赶法进行数值求解. 其它边界条件也可以转化为线性方程组来求解.

2. 算法通用程序

我们给出具有一阶导数边界条件的三阶样条插值的 MATLAB 通用程序. 分为 4 步: (1) 由 (4.32) 式计算 λ_i, μ_i, θ_i 等辅助量; (2) 用追赶法求解 (4.34) 得 $m_1, \cdots, m_{n-1}$; (3) 判断插值点所在区间; (4) 用 (4.29) 计算插值. 下列程序中包含了 5 个子函数, 一个是追赶法, 另外四个是基函数.

```
%maspline.m
function m=maspline(x,y,dy0,dyn,xx)
%用途: 三阶样条插值(一阶导数边界条件)
%格式: m=maspline(x,y,dy0,dyn,xx), x为节点向量,
%y为数据, dy0,dyn为左右两端点的一阶导数如果xx缺省,
%则输出各节点的一阶导数值, m为xx的三阶样条插值
n=length(x)-1;  % 计算小区间的个数
h=diff(x); lambda=h(2:n)./(h(1:n-1)+h(2:n)); mu=1-lambda;
theta=3*(lambda.*diff(y(1:n))./h(1:n-1)+mu.*diff(y(2:n+1))./h(2:n));
theta(1)=theta(1)-lambda(1)*dy0;
theta(n-1)=theta(n-1)-lambda(n-1)*dyn;
% 追赶法解三对角方程组
dy=machase(lambda,2*ones(1:n-1),mu,theta);
% 若给插值点, 计算插值
m=[dy0;dy;dyn];
if nargin>=5
  s=zeros(size(xx));
  for i=1:n
     if i==1
       kk=find(xx<=x(2));
```

```
        else if i==n
          kk=find(xx>x(n));
        else
          kk=find(xx>x(i)&xx<=x(i+1));
        end
        xbar=(xx(kk)-x(i))/h(i);
        s(kk)=alpha0(xbar)*y(i)+alpha1(xbar)*y(i+1)...
           +h(i)*beta0(xbar)*m(i)+h(i)*beta1(xbar)*m(i+1);
    end
    m=s;
end
% 追赶法
function x=machase(a,b,c,d)
n=length(a);
for k=2:n
   b(k)=b(k)-a(k)/b(k-1)*c(k-1);
   d(k)=d(k)-a(k)/b(k-1)*d(k-1);
end
x(n)=d(n)/b(n);
for k=n-1:-1:1
   x(k)=(d(k)-c(k)*x(k+1))/b(k);
end
x=x(:);
% 基函数
function y=alpha0(x)
y=2*x.^3-3*x.^2+1;
function y=alpha1(x)
y=-2*x.^3+3*x.^2;
function y=beta0(x)
y=x.^3-2*x.^2+x;
function y=beta1(x)
y=x.^3-x.^2;
```

例 4.8 利用程序 maspline.m, 求满足下列数据的三阶样条插值

x	-1	0	1	2
$f(x)$	-1	0	1	0
$f'(x)$	0			-1

其中, 插值点为 -0.8, -0.3, 0.2, 0.7, 1.2, 1.7, 2.1.

解 在 MATLAB 命令窗口执行

```
>> x=[-1 0 1 2]; y=[-1 0 1 0];
>> xx=[-0.8 -0.3 0.2 0.7 1.2 1.7 2.1];
>> maspline(x,y,0,-1,xx)
ans =
      -0.9451  -0.4414  0.3045  0.9002  0.9109  0.3546  -0.0905
```

4.4 最小二乘拟合

4.4.1 最小二乘法

前面介绍的插值法, 要求插值函数和被插函数在节点处的函数值甚至导数值完全相同, 这实际上是假定了已知数据相当准确. 但在实际问题中, 数据由观测得到, 难免带有误差. 此时采用高阶插值多项式, 近似程度不一定很好, 有时还会出现 Runge 现象. 所以最好采用最小二乘法.

假定通过观测得到函数 $y=f(x)$ 的 m 个函数值:

$$y_i \approx f(x_i), \quad i=1,2,\cdots,m.$$

所谓最小二乘法就是求 $f(x)$ 的简单近似式 $\varphi(x)$, 使 $\varphi(x_i)$ 与 y_i 的差 (称为残差或偏差)

$$e_i=\varphi(x_i)-y_i, \quad i=1,2,\cdots,m$$

的平方和最小, 即使

$$S=\sum_{i=1}^{m} e_i^2=\sum_{i=1}^{m}[\varphi(x_i)-y_i]^2 \tag{4.37}$$

最小. $\varphi(x)$ 称为 m 个数据 (x_i,y_i), $i=1,2,\cdots,m$ 的最小二乘拟合函数, $f(x)$ 称为被拟合函数. $y\approx\varphi(x)$ 近似反映了变量 x 与 y 之间的函数关系 $y=f(x)$, 称为经验公式或数学模型.

例 4.9 (线性拟合) 已知 $x_1,x_2,\cdots,x_n$ 及 $y_i=f(x_i)\,(i=1,2,\cdots,n)$, 由最小二乘法求 $f(x)$ 的拟合直线 $\varphi(x)=a+bx$.

解 记

$$S(a,b)=\sum_{i=1}^{n}[y_i-\varphi(x_i)]^2=\sum_{i=1}^{n}[y_i-(a+bx_i)]^2.$$

由取极值的必要条件

$$\frac{\partial S}{\partial a}=\frac{\partial S}{\partial b}=0,$$

得

$$\begin{cases}-2\sum_{i=1}^{n}[y_i-(a+bx_i)]=0,\\ -2\sum_{i=1}^{n}x_i[y_i-(a+bx_i)]=0,\end{cases}$$

即

$$\begin{cases}na+\left(\sum_{i=1}^{n}x_i\right)b=\sum_{i=1}^{n}y_i,\\ \left(\sum_{i=1}^{n}x_i\right)a+\left(\sum_{i=1}^{n}x_i^2\right)b=\sum_{i=1}^{n}x_iy_i.\end{cases} \tag{4.38}$$

当 $n>1$ 时, (4.38) 的系数行列式

$$D=\begin{vmatrix} n & \sum_{i=1}^{n}x_i \\ \sum_{i=1}^{n}x_i & \sum_{i=1}^{n}x_i^2 \end{vmatrix}=n\sum_{i=1}^{n}x_i^2-\left(\sum_{i=1}^{n}x_i\right)^2=n\sum_{i=1}^{n}(x_i-\bar{x})^2\neq 0,$$

其中

$$\bar{x}=\frac{1}{n}\sum_{i=1}^{n}x_i.$$

从而 (4.38) 有唯一解.

例 4.10 （线性化拟合）已知 $x_1,x_2,\cdots,x_n$ 及 $y_i=f(x_i)\,(i=1,2,\cdots,n)$, 由最小二乘法求 $f(x)$ 的拟合曲线 $\varphi(x)=ae^{bx}$.

解 这里若与例 4.9 一样, 记 $S(a,b)=\sum_{i=1}^{n}[y_i-\varphi(x_i)]^2$, 则由取极值的必要条件 $S'_a(a,b)=S'_b(a,b)=0$ 得到一个非线性方程组, 难以求解. 为此, 考虑用对数将"曲线拉直". 记

$$z_i=\ln y_i\,(i=1,\cdots,n),\quad \psi(x)=\ln\varphi(x)=\bar{a}+bx\ (\bar{a}=\ln a),$$

则可用 (4.38) 求得 $\bar{a}$ 及 b, 从而

$$\varphi(x)=e^{\psi(x)}=ae^{bx},\quad a=e^{\bar{a}}.$$

例 4.11 当线性方程组未知数的个数少于方程的个数时, 称之为超定方程组. 用最小二乘法求下列超定方程组的数值解:

$$\begin{cases} 4x_1 + 2x_2 = 2, \\ 3x_1 - x_2 = 10, \\ 11x_1 + 3x_2 = 8. \end{cases}$$

解 由最小二乘原理, 即求 x_1, x_2 使下列函数

$$S(x_1, x_2) = (4x_1 + 2x_2 - 2)^2 + (3x_1 - x_2 - 10)^2 + (11x_1 + 3x_2 - 8)^2$$

取极小值. 由

$$\frac{\partial S}{\partial x_1} = \frac{\partial S}{\partial x_2} = 0 \quad \Rightarrow \quad \begin{cases} 73x_1 + 19x_2 = 63, \\ 19x_1 + 7x_2 = 9, \end{cases} \quad \Rightarrow \quad \begin{cases} x_1 = 1.8, \\ x_2 = -3.6. \end{cases}$$

这里, 所得 x_1, x_2 虽非方程组的解, 但却是最小二乘意义下的最佳近似解.

值得说明的是, 上述例子都只是通过取极值的必要条件求出了误差函数的稳定点, 并没有证明它们就是所求的最小值点. 下面我们建立最小二乘拟合的一般理论.

4.4.2 法方程组

首先我们给出函数线性无关的概念.

定义 4.2 设有函数列 $\varphi_0(x), \varphi_1(x), \cdots, \varphi_m(x)$, 如果

$$l_0\varphi_0(x_i) + l_1\varphi_1(x_i) + \cdots + l_m\varphi_m(x_i) = 0, \quad i = 1, 2, \cdots, n \tag{4.39}$$

当且仅当 $l_0 = l_1 = \cdots = l_m = 0$ 时成立, 则称函数 $\varphi_0(x), \varphi_1(x), \cdots, \varphi_m(x)$ 关于节点 $x_1, x_2, \cdots, x_n$ 是线性无关的.

线性无关函数 $\varphi_0, \varphi_1, \cdots, \varphi_m$ 的线性组合全体 Φ 称为由 $\varphi_0, \varphi_1, \cdots, \varphi_m$ 张成的函数空间, 记为

$$\Phi = \text{span}\{\varphi_0, \varphi_1, \cdots, \varphi_m\} = \left\{\varphi(x) = \sum_{i=0}^{m} a_i\varphi_i \middle| a_0, a_1, \cdots, a_m \in \mathbf{R}\right\}.$$

并称 $\varphi_0, \varphi_1, \cdots, \varphi_m$ 为 Φ 的基函数.

最小二乘拟合用数学语言表述为：已知数据 x_i, $y_i = f(x_i)\,(i = 1, 2, \cdots, n)$ 和函数空间

$$\Phi = \text{span}\{\varphi_0, \varphi_1, \cdots, \varphi_m\},$$

求一函数 φ^*, 使

$$\|f - \varphi^*\|_2 = \min_{\varphi \in \Phi} \|f - \varphi\|_2.$$

令

$$\varphi(x)=\sum_{j=0}^{m}a_j\varphi_j(x),\quad \varphi^*(x)=\sum_{j=0}^{m}a_j^*\varphi_j(x),$$

那么

$$S(a_0,a_1,\cdots,a_m)=\|f-\varphi\|_2^2=\sum_{i=1}^{n}\left[y_i-\sum_{j=0}^{m}a_j\varphi_j(x_i)\right]^2. \tag{4.40}$$

于是, 问题等价于求 $a_0^*,a_1^*,\cdots,a_m^*\in\mathbf{R}$, 使

$$S(a_0^*,a_1^*,\cdots,a_m^*)=\min_{a_0,a_1,\cdots,a_m\in\mathbf{R}}S(a_0,a_1,\cdots,a_m). \tag{4.41}$$

根据函数极值的必要条件, 对 $a_0,a_1,\cdots,a_m$ 求偏导数并令其等于零:

$$\frac{\partial S}{\partial a_k}=0,\quad k=0,1,\cdots,m,$$

得

$$-2\sum_{i=1}^{n}\left[y_i-\sum_{j=0}^{m}a_j\varphi_j(x_i)\right]\varphi_k(x_i)=0,$$

即

$$\sum_{j=0}^{m}\sum_{i=1}^{n}a_j\varphi_j(x_i)\varphi_k(x_i)=\sum_{i=1}^{n}y_i\varphi_k(x_i),\quad k=0,1,\cdots,m.$$

用内积表示为线性方程组

$$\sum_{j=0}^{m}(\varphi_j,\varphi_k)a_j=(f,\varphi_k),\quad k=0,1,\cdots,m, \tag{4.42}$$

其矩阵形式为

$$\begin{pmatrix}(\varphi_0,\varphi_0)&(\varphi_0,\varphi_1)&\cdots&(\varphi_0,\varphi_m)\\(\varphi_1,\varphi_0)&(\varphi_1,\varphi_1)&\cdots&(\varphi_1,\varphi_m)\\\vdots&\vdots&&\vdots\\(\varphi_m,\varphi_0)&(\varphi_m,\varphi_1)&\cdots&(\varphi_m,\varphi_m)\end{pmatrix}\begin{pmatrix}a_0\\a_1\\\vdots\\a_m\end{pmatrix}=\begin{pmatrix}(f,\varphi_0)\\(f,\varphi_1)\\\vdots\\(f,\varphi_m)\end{pmatrix}. \tag{4.43}$$

方程组 (4.43) 称为法方程组或正规方程组.

定理 4.2 如果函数 $\varphi_0(x),\varphi_1(x),\cdots,\varphi_m(x)$ 关于节点 $x_1,x_2,\cdots,x_n$ 线性无关, 则法方程组 (4.43) 的解存在唯一, 且是 (4.41) 的唯一最优解.

证 用 $\varphi_k(x_i)$ 乘 (4.39) 的两边并求和得

$$l_0(\varphi_0,\varphi_k)+l_1(\varphi_1,\varphi_k)+\cdots+l_m(\varphi_m,\varphi_k)=0,\ \ k=0,1,\cdots,m. \tag{4.44}$$

由于函数 $\varphi_0(x), \varphi_1(x), \cdots, \varphi_m(x)$ 关于节点 $x_1, x_2, \cdots, x_n$ 线性无关, 故 (4.44) 只有零解, 那么必有

$$\begin{vmatrix} (\varphi_0,\varphi_0) & (\varphi_0,\varphi_1) & \cdots & (\varphi_0,\varphi_m) \\ (\varphi_1,\varphi_0) & (\varphi_1,\varphi_1) & \cdots & (\varphi_1,\varphi_m) \\ \vdots & \vdots & & \vdots \\ (\varphi_m,\varphi_0) & (\varphi_m,\varphi_1) & \cdots & (\varphi_m,\varphi_m) \end{vmatrix} \neq 0,$$

这样, 法方程组 (4.43) 的解存在唯一.

下面证明 (4.41). 设 $a_0^*, a_1^*, \cdots, a_m^*$ 是法方程组的解:

$$\sum_{j=0}^{m} (\varphi_j, \varphi_k) a_j^* = (f, \varphi_k), \quad k = 0, 1, \cdots, m,$$

即

$$(\varphi^*, \varphi_k) = (f, \varphi_k), \quad k = 0, 1, \cdots, m,$$

或

$$(f - \varphi^*, \varphi_k) = 0, \quad k = 0, 1, \cdots, m,$$

根据内积的性质, 对任意的 $\varphi \in \Phi$ 有

$$(f - \varphi^*, \varphi) = 0.$$

于是, 对任意的 $a_0, a_1, \cdots, a_m$, 有

$$\begin{aligned} & S(a_0, a_1, \cdots, a_m) = \|f - \varphi\|_2^2 \\ = & (f - \varphi, f - \varphi) = (f - \varphi^* + \varphi^* - \varphi, f - \varphi^* + \varphi^* - \varphi) \\ = & (f - \varphi^*, f - \varphi^*) + 2(f - \varphi^*, \varphi^* - \varphi) + (\varphi^* - \varphi, \varphi^* - \varphi) \\ = & S(a_0^*, a_1^*, \cdots, a_m^*) + 2(f - \varphi^*, \varphi^* - \varphi) + \|\varphi^* - \varphi\|_2^2, \end{aligned}$$

由于 $\varphi^* - \varphi \in \Phi$, 因此, 上面最后一个等式的右边第二项为零, 而第三项非负, 故

$$S(a_0, a_1, \cdots, a_m) \geqslant S(a_0^*, a_1^*, \cdots, a_m^*),$$

即 $a_0^*, a_1^*, \cdots, a_m^*$ 是 (4.41) 的唯一最优解. □

例 4.12 已知 $\sin 0 = 0$, $\sin\dfrac{\pi}{6} = \dfrac{1}{2}$, $\sin\dfrac{\pi}{3} = \dfrac{\sqrt{3}}{2}$, $\sin\dfrac{\pi}{2} = 1$, 由最小二乘法求 $\sin x$ 的拟合曲线 $\varphi(x) = ax + bx^3$.

解 这里, $f(x)=\sin x$, $\varphi_0(x)=x$, $\varphi_1=x^3$, 计算得

$$(\varphi_0,\varphi_0)=\sum_{i=1}^{4}[\varphi_0(x_i)]^2=\sum_{i=1}^{4}x_i^2=3.8382,$$

$$(\varphi_0,\varphi_1)=(\varphi_1,\varphi_0)=\sum_{i=1}^{4}\varphi_0(x_i)\varphi_1(x_i)=\sum_{i=1}^{4}x_i^4=7.3658,$$

$$(\varphi_1,\varphi_1)=\sum_{i=1}^{4}[\varphi_1(x_i)]^2=\sum_{i=1}^{4}x_i^6=16.3611,$$

$$(f,\varphi_0)=\sum_{i=1}^{4}x_i\sin x_i=2.7395,\quad (f,\varphi_1)=\sum_{i=1}^{4}x_i^3\sin x_i=4.9421.$$

得法方程组

$$\begin{cases}3.8382a+7.3685b=2.7395,\\ 7.3658a+16.3611b=4.9421,\end{cases}$$

解得 $a=0.9856$, $b=-0.1417$. 从而对应已知数据的 $\sin x$ 的最小二乘拟合曲线为

$$\varphi(x)=0.9856x-0.1417x^3.$$

4.4.3 正交最小二乘拟合

最常见的拟合函数类是多项式, 其基函数一般取幂函数

$$\varphi_0(x)=1,\ \varphi_1(x)=x,\ \cdots,\varphi_m(x)=x^m.$$

由于

$$(\varphi_j,\varphi_k)=\sum_{i=1}^{n}x_i^{j+k},\quad (f,\varphi_k)=\sum_{i=1}^{n}x_i^k y_i,$$

这样, 法方程组为

$$\begin{pmatrix} n & \sum_{i=1}^{n}x_i & \cdots & \sum_{i=1}^{n}x_i^m \\ \sum_{i=1}^{n}x_i & \sum_{i=1}^{n}x_i^2 & \cdots & \sum_{i=1}^{n}x_i^{m+1} \\ \vdots & \vdots & & \vdots \\ \sum_{i=1}^{n}x_i^m & \sum_{i=1}^{n}x_i^{m+1} & \cdots & \sum_{i=1}^{n}x_i^{2m}\end{pmatrix}\begin{pmatrix}a_0\\ a_1\\ \vdots\\ a_m\end{pmatrix}=\begin{pmatrix}\sum_{i=1}^{n}y_i\\ \sum_{i=1}^{n}x_iy_i\\ \vdots\\ \sum_{i=1}^{n}x_i^m y_i\end{pmatrix}. \tag{4.45}$$

但遗憾的是, 当 m 比较大时, 该方程组往往是病态的, 从而导致结果误差很大.

下面我们考虑所谓的正交最小二乘拟合. 首先给出正交多项式的概念.

定义 4.3 设节点 $x_1, x_2, \cdots, x_n$ 和多项式函数 $P(x)$ 和 $Q(x)$, 如果

$$(P, Q) = \sum_{i=1}^{n} P(x_i) Q(x_i) = 0,$$

则称 $P(x)$ 和 $Q(x)$ 关于节点 $x_1, x_2, \cdots, x_n$ 正交. 如果函数类 Φ 的基函数 $\psi_0, \psi_1, \cdots, \psi_m$ 两两正交, 则称为一组正交基.

设 $\psi_0, \psi_1, \cdots, \psi_m$ 为函数类 Φ 的一组正交基, 那么法方程组 (4.43) 就成为简单的对角方程组, 其解可以有下式直接给出:

$$a_k = \frac{(f, \psi_k)}{(\psi_k, \psi_k)}, \quad k = 0, 1, \cdots, m, \tag{4.46}$$

从而避免了求解病态方程组.

正交基可以由任意基 $\varphi_0, \varphi_1, \cdots, \varphi_m$ 通过 Schmit 正交化方法得到:

$$\psi_0(x) = \varphi_0(x),$$

$$\psi_1(x) = \varphi_1(x) - \frac{(\varphi_1, \psi_0)}{(\psi_0, \psi_0)} \psi_0(x),$$

$$\psi_2(x) = \varphi_2(x) - \frac{(\varphi_2, \psi_0)}{(\psi_0, \psi_0)} \psi_0(x) - \frac{(\varphi_2, \psi_1)}{(\psi_1, \psi_1)} \psi_1(x),$$

$$\cdots\cdots\cdots\cdots$$

$$\psi_m(x) = \varphi_m(x) - \frac{(\varphi_m, \psi_0)}{(\psi_0, \psi_0)} \psi_0(x) - \frac{(\varphi_m, \psi_1)}{(\psi_1, \psi_1)} \psi_1(x) - \cdots - \frac{(\varphi_m, \psi_{m-1})}{(\psi_{m-1}, \psi_{m-1})} \psi_{m-1}(x).$$

例 4.13 已知下列数据求拟合曲线 $\varphi(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$.

x	-2	-1	0	1	2
$f(x)$	-1	-1	0	1	1

解 取 $\varphi_0(x) = 1$, $\varphi_1(x) = x$, $\varphi_2(x) = x^2$, $\varphi_3(x) = x^3$, 先进行 Schmit 正交化:

$$\psi_0(x) = \varphi_0(x) = 1,$$

$$\psi_1(x) = \varphi_1(x) - \frac{(\varphi_1, \psi_0)}{(\psi_0, \psi_0)} \psi_0(x) = x,$$

$$\psi_2(x) = \varphi_2(x) - \frac{(\varphi_2, \psi_0)}{(\psi_0, \psi_0)} \psi_0(x) - \frac{(\varphi_2, \psi_1)}{(\psi_1, \psi_1)} \psi_1(x) = x^2 - 2,$$

$$\psi_3(x) = \varphi_3(x) - \frac{(\varphi_3, \psi_0)}{(\psi_0, \psi_0)} \psi_0(x) - \frac{(\varphi_3, \psi_1)}{(\psi_1, \psi_1)} \psi_1(x) - \frac{(\varphi_3, \psi_2)}{(\psi_2, \psi_2)} \psi_2(x) = x^3 - \frac{17}{5} x.$$

则 $\psi_0, \psi_1, \psi_2, \psi_3$ 两两正交. 计算得

$$(\psi_0, \psi_0) = 5, \quad (\psi_1, \psi_1) = 10, \quad (\psi_2, \psi_2) = 14, \quad (\psi_3, \psi_3) = 14.4,$$
$$(f, \psi_0) = 0, \quad (f, \psi_1) = 6, \quad (f, \psi_2) = 0, \quad (f, \psi_3) = -2.4.$$

从而, 由 (4.46) 得

$$a_0 = 0, \ \ a_1 = \frac{6}{10} = \frac{3}{5}, \ \ a_2 = 0, \ \ a_3 = \frac{-2.4}{14.4} = -\frac{1}{6}.$$

故

$$\varphi(x) = \frac{3}{5}\psi_1(x) - \frac{1}{6}\psi_3(x) = \frac{3}{5}x - \frac{1}{6}\left(x^3 - \frac{17}{5}x\right) = \frac{7}{6}x - \frac{1}{6}x^3.$$

4.4.4 多项式拟合的通用程序

下面给出多项式拟合的 MATLAB 通用程序:

- 多项式拟合 MATLAB 程序

```
%mafit.m
function p=mafit(x,y,m)
%用途: 多项式拟合
%格式: p=mafit(x,y,m)  x,y为数据向量, m为拟合
%多项式次数, p返回多项式系数降幂排列
format short;
A=zeros(m+1,m+1);
for i=0:m
  for j=0:m
    A(i+1,j+1)=sum(x.^(i+j));
  end
  b(i+1)=sum(x.^i.*y);
end
a=A\b';
p=fliplr(a');  %按降幂排列
```

例 4.14 用上述程序求解例 4.13.

解 在 MATLAB 命令窗口执行

```
>> x=-2:2; y=[-1 -1 0 1 1];
>> p=mafit(x,y,3)
```

得计算结果:

```
p =
      -0.1667      0      1.1667      0
```

从而所求的拟合曲线为

$$\varphi(x) = -0.1667x^3 + 1.1667x.$$

习 题 4

(Ⅰ) 理论分析题

4.1 当 $x=-1,1,2$ 时, $f(x)=-3,0,4$, 求 $f(x)$ 的拉格朗日插值多项式.

4.2 利用函数 $y=\sqrt{x}$ 在 $x=1,4$ 的值, 计算 $\sqrt{2}$ 的近似值, 并估计误差.

4.3 利用函数 $y=\sqrt{x}$ 在 $x=0,1,4$ 的值, 计算 $\sqrt{2}$ 的近似值, 并估计误差.

4.4 设被插函数为 $y=2^x$.

(1) 给定两点坐标 $(0,1)$, $(1,2)$, 构造线性插值函数 $L_1(x)$, 并求 $2^{0.3}$ 的近似值和截断误差;

(2) 给定三点坐标 $(-1.0.5)$, $(0,1)$, $(1,2)$, 构造二次插值函数 $L_2(x)$, 并求 $2^{0.3}$ 的近似值和截断误差.

4.5 已知函数 $f(x)=56x^3+24x^2+5$ 在点 2^0, 2^1, 2^5, 2^7 的函数值, 求其三次插值多项式.

4.6 证明: (1) $\sum\limits_{k=0}^{n} x_k^j l_k(x) = x^j,\quad j=1,2,\cdots,n$; (2) $\sum\limits_{k=0}^{n}(x_k-x)^j l_k(x)=0,\quad j=1,2,\cdots,n$.

4.7 设 $f(x)$ 在 $[a,\ b]$ 有连续的二阶导数, 且 $f(a)=f(b)=0$, 求证

$$\max_{a\leqslant x\leqslant b}|f(x)| \leqslant \frac{1}{8}(b-a)^2 \max_{a\leqslant x\leqslant b}|f''(x)|.$$

4.8 设 $f(x)=x^4$, 用拉格朗日余项定理写出以 $-1,0,1,3$ 为节点的三次差值多项式.

4.9 利用余项定理证明次数 $\leqslant n$ 的多项式, 其 n 次拉格朗日差值多项式就是它自身.

4.10 设 $f(x)\in C^1[a,\ b]$, $x_0\in(a,\ b)$, 定义 $f[x_0,\ x_0]=\lim\limits_{x\to x_0}[x,\ x_0]$, 证明 $f[x_0,\ x_0]=f'(x_0)$.

4.11 若 $f(x)=\omega_{n+1}(x)=(x-x_0)(x-x_1)\cdots(x-x_n)$, $x_i(i=0,1,\cdots,n)$ 互异, 求 $f[x_0,\ x_1,\cdots,x_p]$ 的值, 这里 $p\leqslant n+1$.

4.12 已知 $\sin(0.32)=0.314567$, $\sin(0.34)=0.333487$ 有 6 位有效数字.

(1) 用拉格朗日插值多项式求 $\sin(0.33)$ 的近似值;

(2) 证明在区间 $[0.32, 0.34]$ 上用拉格朗日插值多项式计算 $\sin x$ 时至少有 4 位有效数字.

4.13 设 $f(x)$ 在 $[1,3]$ 上 4 阶可导, $P(x)$ 为满足 $P(1)=f(1)$, $P(2)=f(2)$, $P(3)=f(3)$ 及 $P'(2)=f'(2)$ 的不超过 3 次的多项式, 证明: 当 $x\in[1,3]$ 时, 存在 $\xi\in[1.3]$, 使得

$$f(x)-P(x)=\frac{f^{(4)}(\xi)}{4!}(x-1)(x-2)^2(x-3).$$

4.14 求不超过 4 次的多项式 $P(x)$, 使满足 $P(0) = P'(0) = 0$, $P(1) = P'(1) = 1$, $P(2) = 1$, 并当 $f(x)$ 也满足上述关于 $P(x)$ 的约束时, 写出余项 $f(x) - P(x)$.

4.15 给定数据如下：

x_i	0.5	1.0	1.5	2.0
$f(x_i)$	0.75	1.25	2.50	5.50

(1) 作函数 $f(x)$ 的差商表;

(2) 用牛顿插值公式求三次插值多项式 $N_3(x)$.

4.16 设 $x_0, x_1, \cdots, x_n$ 是互异的插值节点, 试证：

(1) $\sum\limits_{i=0}^{n} x_i^k l_i(x) = x^k \ (k = 0, 1, \cdots, n)$, 其中 $l_i(x)$ 是拉格朗日插值基函数;

(2) 若 $f(x)$ 为不超过 n 次的多项式, 则有 $L_n(x) = f(x)$, 其中 $L_n(x)$ 是 $f(x)$ 的以 $x_0, x_1, \cdots, x_n$ 为插值节点的拉格朗日插值多项式.

4.17 设有 $n+1$ 个多项式

$$\varphi_k(x) = \sum_{i=0}^{k} c_i^{(k)} x^i, \quad c_i^{(k)} \neq 0, \quad k = 0, 1, \cdots, n.$$

试证明 $\varphi_0(x), \varphi_1(x), \cdots, \varphi_n(x)$ 在任何区间 $[a, b]$ 上都线性无关.

4.18 设 $P(x)$ 是任意首项系数为 1 的 $n+1$ 次多项式, 试证明

$$P(x) - \sum_{k=0}^{n} P(x_k) l_k(x) = \omega(x),$$

其中 $l_k(x)\,(k = 0, 1, \cdots, n)$ 是拉格朗日插值基函数, $\omega(x) = (x - x_0)(x - x_1) \cdots (x - x_n)$.

4.19 求被插函数 $f(x)$ 在区间 $[1, 5]$ 上的三次样条函数 $S(x)$, 其中 $f(1) = 1$, $f(2) = 3$, $f(4) = 4$, $f(5) = 2$, 取自然边界条件 $S''(1) = S''(5) = 0$.

4.20 求被插函数 $f(x)$ 在区间 $[-1, 2]$ 上的三次样条函数 $S(x)$, 其中 $f(-1) = -1$, $f(0) = 0$, $f(1) = 1$, $f(2) = 0$, 取边界条件 $S'(-1) = 0$, $S''(1) = -1$.

4.21 已知连续函数 $f(x)$ 在 $x = -1,\ 0,\ 2,\ 3$ 的值分别是 $-4, -1, 0, 3$, 用牛顿插值求：

(1) $f(1.5)$ 的近似值;

(2) $f(x) = 0.5$ 时, x 的近似值.

4.22 对如下函数表, 建立三次样条插值函数.

x	1	2	3
$f(x)$	2	4	2
$f'(x)$	1		-1

4.23 求超定方程组

$$\begin{cases} x_2 = 1, \\ x_1 + x_2 = 2.1, \\ 2x_1 + x_2 = 2.9, \\ 3x_1 + x_2 = 3.2 \end{cases}$$

的最小二乘解.

4.24 已知 $\cos\frac{\pi}{6} = \frac{\sqrt{3}}{2}$, $\cos\frac{\pi}{3} = \frac{1}{2}$, $\cos\frac{\pi}{2} = 0$, 由最小二乘法求 $\cos x$ 的拟合曲线, 并比较它们的拟合误差:

(1) $\varphi_1(x) = a + bx$; (2) $\varphi_2(x) = a + bx^2$; (3) $\varphi_3(x) = ae^{bx}$.

4.25 求 $f(x) = x^5 - 1$ 关于节点 $-1, 0, 1, 2, 3$ 的最小二乘三次逼近多项式:

(1) 由基 1, x, x^2, x^3 直接计算; (2) 将基 1, x, x^2, x^3 正交化后计算.

4.26 已知一组实验数据

x_i	1	2	3	4	5
y_i	4	4.5	6	8	8.5
ω_i	2	1	3	1	1

试求最小二乘拟合曲线.

4.27 用最小二乘法求形如 $y = a + bx^2$ 的多项式, 使之与下列数据相拟合.

x	19	25	31	38	44
y	19.0	32.3	49.0	73.3	97.8

4.28 求超定方程组

$$\begin{cases} 2x_1 + 4x_2 = 11, \\ 3x_1 - 5x_2 = 3, \\ x_1 + 2x_2 = 6, \\ 2x_1 + x_2 = 7 \end{cases}$$

的最小二乘解, 并求误差平方和.

(Ⅱ) 上机实验题

4.1 编制拉格朗日插值法 MATLAB 程序, 求 $\ln 0.53$ 的近似值. 已知 $f(x) = \ln x$ 的数值如下表所示:

x	0.4	0.5	0.6	0.7
$\ln x$	−0.916291	−0.693147	−0.510826	−0.357765

4.2 编制牛顿插值法 MATLAB 程序, 求 $f(0.5)$ 的近似值. 已知的数值如下表所示:

x_i	0.0	0.2	0.4	0.6	0.8
$f(x_i)$	0.1995	0.3965	0.5881	0.7721	0.9461

4.3 对于三次样条插值的三弯矩方法, 编制用于第一种和第二种边界条件的 MATLAB 程序. 设已知数据

x_i	0.25	0.30	0.39	0.45	0.53
y_i	0.5000	0.5477	0.6245	0.6708	0.7280

第一种边界条件: $S'(0.25) = 1.000$, $S'(0.53) = 0.6868$;

第二种边界条件: $S''(0.25) = S''(0.53) = 0$.

分别用所编程序求解, 输出各插值节点的弯矩值 $\{m_i\}$ 和插值中点的样条函数值, 并作点列 $\{x_i, y_i\}$ 和样条函数 $y = S(x)$ 的图形.

4.4 编制以函数 $x^k\ (k = 0, 1, \cdots, m)$ 为基的多项式最小二乘拟合 MATLAB 程序, 并用于对下列数据作三次多项式最小二乘拟合:

x_i	−1.0	−0.5	0.0	0.5	1.0	1.5	2.0
y_i	−4.447	−0.452	0.551	0.048	−0.447	0.549	4.552

求拟合曲线 $\varphi(x) = a_0 + a_1x + \cdots + a_nx^n$ 中的参数 $\{a_k\}$、平方误差 δ^2, 并作离散数据 $\{x_i, y_i\}$ 和拟合曲线 $y = \varphi(x)$ 的图形.

第 5 章　数值积分和数值微分

在数学分析中, 积分值是通过找原函数的办法得到的. 我们知道找一个函数的原函数并非一件容易的事情, 许多函数甚至不存在初等函数表示的原函数. 因此有必要研究积分的数值计算问题.

前一章指出, 如果所给函数 $f(x)$ 比较复杂, 可以构造插值多项式 $L_n(x)$ 作为 $f(x)$ 的近似表达式, 然后通过处理 $L_n(x)$ 得到 $f(x)$ 的近似结果. 本章根据这一观点讨论数值积分和数值微分.

5.1　插值型求积公式

我们用插值多项式 $L_n(x)$ 替换积分

$$I^* = \int_a^b f(x)\mathrm{d}x$$

中的被积函数 $f(x)$, 然后计算

$$I = \int_a^b L_n(x)\mathrm{d}x \tag{5.1}$$

作为积分的近似值, 这样建立的求积公式称为插值型求积公式.

用插值多项式 $L_n(x)$ 的表达式

$$L_n(x) = \sum_{k=0}^{n} l_k(x) f(x_k)$$

代入 (5.1) 得

$$I = \sum_{k=0}^{n} A_k f(x_k), \tag{5.2}$$

其中

$$A_k = \int_a^b l_k(x)\mathrm{d}x = \int_a^b \prod_{\substack{j=0\\ j\neq k}}^{n} \frac{x-x_j}{x_k-x_j}\mathrm{d}x \tag{5.3}$$

$$= \int_a^b \frac{(x-x_0)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_n)}{(x_k-x_0)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x_k-x_n)}\mathrm{d}x$$

称为求积系数, 而 $x_k\ (k=0,1,\cdots,n)$ 则称为求积节点.

一般来说, I 的值都不等于 I^* 的精确值, 它们的差称为求积公式 (5.2) 的余项, 也叫做截断误差. 由插值余项公式知, 对于插值型求积公式 (5.3), 其余项为

$$R[f]=I^*-I=\int_a^b \frac{f^{(n+1)}(\xi)}{(n+1)!}\omega(x)\,\mathrm{d}x, \tag{5.4}$$

其中 ξ 与变量 x 有关, 且 $\omega(x)=(x-x_0)(x-x_1)\cdots(x-x_n)$.

通常用所谓的代数精度表示求积公式的误差. 下面给出代数精度的定义.

定义 5.1 *如果求积公式 (5.2) 对于一个不超过 m 次的多项式是准确的, 即 $R[f]=0$; 而对于 $m+1$ 次以上的多项式是不准确的, 则称求积公式 (5.2) 的代数精度为 m.*

利用代数精度的概念, 可以得到下面的结论.

定理 5.1 *求积公式 (5.2) 是插值型求积公式的充要条件是: 它的代数精度至少为 n.*

证 必要性. 若公式 (5.2) 是插值型求积公式, 则 (5.4) 成立. 故对于次数不超过 n 的多项式 $f(x)$, 其余项 $R[f]=0$, 故此时求积公式 (5.2) 的代数精度至少为 n.

充分性. 若求积公式 (5.2) 的代数精度至少为 n, 则它对于插值基函数 $l_k(x)\ (k=0,1,\cdots,n)$ 是准确的, 即有

$$\int_a^b l_k(x)\mathrm{d}x=\sum_{j=0}^{n}A_j l_k(x_j),$$

由于 $l_k(x_j)=\delta_{kj}$, 上式右边实际上就等于 A_k, 即

$$A_k=\int_a^b l_k(x)\mathrm{d}x$$

成立, 故 (5.2) 是插值型求积公式. □

例 5.1 证明下面的求积公式具有 3 次代数精度:

$$\int_0^1 f(x)\mathrm{d}x\approx\frac{1}{2}[f(0)+f(1)]-\frac{1}{12}[f'(1)-f'(0)]. \tag{5.5}$$

证 (1) 令 $f(x)=1$, 代入求积公式 (5.5) 得, 左边 $=1=$ 右边.

(2) 令 $f(x)=x$, 代入求积公式 (5.5) 得,

$$\text{左边}=\int_0^1 x\mathrm{d}x=\frac{1}{2}\ =\ \text{右边}=\frac{1}{2}(0+1)-\frac{1}{12}(1-1)=\frac{1}{2}.$$

(3) 令 $f(x)=x^2$, 代入求积公式 (5.5) 得,

$$\text{左边}=\int_0^1 x^2\mathrm{d}x=\frac{1}{3}\ =\ \text{右边}=\frac{1}{2}(0+1)-\frac{1}{12}(2-0)=\frac{1}{2}-\frac{1}{6}=\frac{1}{3}.$$

(4) 令 $f(x)=x^3$, 代入求积公式 (5.5) 得,

$$\text{左边}=\int_0^1 x^3\mathrm{d}x=\frac{1}{4}=\text{右边}=\frac{1}{2}(0+1)-\frac{1}{12}(3-0)=\frac{1}{2}-\frac{1}{4}=\frac{1}{4}.$$

(5) 令 $f(x)=x^4$, 代入求积公式 (5.5) 得

$$\text{左边}=\int_0^1 x^4\mathrm{d}x=\frac{1}{5}\neq\text{右边}=\frac{1}{2}(0+1)-\frac{1}{12}(4-0)=\frac{1}{2}-\frac{1}{3}=\frac{1}{6}.$$

上述表明, 求积公式 (5.5) 对不超过 3 次的多项式是准确的, 而对 3 次以上的多项式是不准确的. 根据代数精度的定义, 求积公式 (5.5) 具有 3 次代数精度. □

下面我们来讨论求积系数 A_k 的计算. 为了方便, 我们取等距节点, 即把积分区间 $[a,b]$ 剖分成 n 等分. 令步长 $h=(b-a)/n$, 并记 $x_0=a,\ x_n=b$, 则 $n+1$ 个节点为

$$x_k=x_0+kh,\quad k=0,1,\cdots,n.$$

作变换

$$t=\frac{x-x_0}{h},$$

代入求积系数公式 (5.3) 得

$$\begin{aligned}A_k&=\int_a^b\frac{(x-x_0)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_n)}{(x_k-x_0)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x_k-x_n)}\mathrm{d}x\\&=\int_0^n\frac{h^n t(t-1)\cdots(t-k+1)(t-k-1)\cdots(t-n)}{(-1)^{n-k}h^n(n-k)!\,k!}h\mathrm{d}t\qquad(5.6)\\&=\frac{(-1)^{n-k}h}{(n-k)!\,k!}\int_0^n t(t-1)\cdots(t-k+1)(t-k-1)\cdots(t-n)\mathrm{d}t.\end{aligned}$$

这种等距节点的插值型求积公式通常称为牛顿–科茨公式. 下面介绍几个牛顿–科茨公式的特殊形式并分析其截断误差.

5.2 几个常用的求积公式

5.2.1 梯形公式及其误差

利用牛顿–科茨公式, 取 $n=1$, 即 $x_0=a,\ x_1=b$, 此时, $h=b-a$, 代入 (5.6), 计算得

$$A_0=\frac{(-1)^1h}{1!\,0!}\int_0^1(t-1)\mathrm{d}t=-\frac{1}{2}[(1-1)^2-(0-1)^2]h=\frac{b-a}{2},$$

$$A_1=\frac{(-1)^1h}{0!\,1!}\int_0^1 t\,\mathrm{d}t=\frac{1}{2}(1^2-0^2)h=\frac{b-a}{2}.$$

所以梯形公式为

$$T=\frac{b-a}{2}\big[f(a)+f(b)\big]. \tag{5.7}$$

由 (5.4), 梯形公式 (5.7) 的误差为:

$$R_T[f]=-\frac{(b-a)^3}{12}f''(\eta),\quad a<\eta<b. \tag{5.8}$$

从上述余项公式可以看出, 梯形求积公式的代数精度为 1.

例 5.2 利用梯形公式计算

$$I=\int_0^1\frac{4}{1+x^2}\mathrm{d}x$$

的近似值.

解 由梯形公式 (5.7), 有

$$\int_0^1\frac{4}{1+x^2}\mathrm{d}x\approx\frac{1-0}{2}\Big(\frac{4}{1+0^2}+\frac{4}{1+1^2}\Big)=3.$$

5.2.2 辛普森公式及其误差

利用牛顿–科茨公式, 取 $n=2$, 即 $x_0=a$, $x_1=(a+b)/2$, $x_2=b$, 此时, $h=(b-a)/2$, 代入 (5.6) 计算得

$$A_0=\frac{(-1)^2h}{2!\,0!}\int_0^2(t-1)(t-2)\mathrm{d}t=\frac{h}{2}\times\frac{2}{3}=\frac{h}{3}=\frac{b-a}{6},$$

$$A_1=\frac{(-1)^1h}{1!\,1!}\int_0^2 t(t-2)\mathrm{d}t=-h\times\Big(-\frac{4}{3}\Big)=\frac{4h}{3}=\frac{4(b-a)}{6},$$

$$A_2=\frac{(-1)^0h}{0!\,2!}\int_0^2 t(t-1)\mathrm{d}t=\frac{h}{2}\times\frac{2}{3}=\frac{h}{3}=\frac{b-a}{6}.$$

故辛普森公式为

$$S=\frac{b-a}{6}\big[f(a)+4f(c)+f(b)\big], \tag{5.9}$$

其中 $c=(a+b)/2$ 为区间 $[a,b]$ 的中点. 辛普森公式通常也称为抛物形公式.

可以证明, 辛普森公式 (5.9) 的误差为

$$R_S[f]=-\frac{1}{90}\Big(\frac{b-a}{2}\Big)^5f^{(4)}(\eta),\quad a<\eta<b. \tag{5.10}$$

事实上, 假设 $f^{(4)}(x)$ 在 $[a,b]$ 上近似地取定值 C_4, 将 $f(x)$ 在 $[a,b]$ 的中点 $c=(a+b)/2$ 处泰勒展开

$$f(x)=f(c)+f'(c)(x-c)+\frac{f''(c)}{2!}(x-c)^2+\frac{f^{(3)}(c)}{3!}(x-c)^3+\frac{C_4}{4!}(x-c)^4. \tag{5.11}$$

然后将该展开式在 $[a,b]$ 上积分，注意到函数 $x-c$ 和 $(x-c)^3$ 在 $[a,b]$ 上的积分为 0，故右端第二项和第四项的积分值均为 0，于是我们有

$$I^* = \int_a^b f(x)\mathrm{d}x \approx f(c)(b-a) + \frac{f''(c)}{3}\left(\frac{b-a}{2}\right)^3 + \frac{C_4}{60}\left(\frac{b-a}{2}\right)^5. \tag{5.12}$$

另一方面，在 (5.11) 中分别令 $x=a$ 和 $x=b$ 得

$$f(a) = f(c) - f'(c)\frac{b-a}{2} + \frac{f''(c)}{2!}\left(\frac{b-a}{2}\right)^2 - \frac{f^{(3)}(c)}{3!}\left(\frac{b-a}{2}\right)^3 + \frac{C_4}{4!}\left(\frac{b-a}{2}\right)^4,$$

$$f(b) = f(c) + f'(c)\frac{b-a}{2} + \frac{f''(c)}{2!}\left(\frac{b-a}{2}\right)^2 + \frac{f^{(3)}(c)}{3!}\left(\frac{b-a}{2}\right)^3 + \frac{C_4}{4!}\left(\frac{b-a}{2}\right)^4.$$

代入辛普森公式 (5.9) 得

$$S \approx f(c)(b-a) + \frac{f''(c)}{3}\left(\frac{b-a}{2}\right)^3 + \frac{C_4}{36}\left(\frac{b-a}{2}\right)^5.$$

于是利用 (5.12)，有

$$R_S[f] = I^* - S \approx -\frac{C_4}{90}\left(\frac{b-a}{2}\right)^5 = -\frac{1}{90}\left(\frac{b-a}{2}\right)^5 f^{(4)}(\eta), \quad a<\eta<b.$$

由余项公式 (5.10) 可知，辛普森公式的代数精度为 3.

例 5.3 利用辛普森公式计算

$$I = \int_0^1 \frac{4}{1+x^2}\mathrm{d}x$$

的近似值.

解 由辛普森公式 (5.9)，有

$$\int_0^1 \frac{4}{1+x^2}\mathrm{d}x \approx \frac{1-0}{6}\left(\frac{4}{1+0^2} + 4\times\frac{4}{1+0.5^2} + \frac{4}{1+1^2}\right) = 3.1333.$$

5.2.3 科茨公式及其误差

利用牛顿 - 科茨公式，取 $n=4$，则 $h=(b-a)/4$，于是求积节点为 $x_0=a$，$x_1=d=x_0+h$，$x_2=c=x_0+2h$，$x_3=e=x_0+3h$，$x_4=b$，代入 (5.6) 计算得

$$A_0 = A_4 = \frac{7}{90}(b-a), \quad A_1 = A_3 = \frac{32}{90}(b-a), \quad A_2 = \frac{12}{90}(b-a).$$

故科茨公式为

$$C = \frac{b-a}{90}\left[7f(a) + 32f(d) + 12f(c) + 32f(e) + 7f(b)\right]. \tag{5.13}$$

科茨公式 (5.13) 也称为布尔公式. 可以证明, 科茨公式的余项为

$$R_C[f] = -\frac{8}{945}\left(\frac{b-a}{4}\right)^7 f^{(6)}(\eta), \quad a < \eta < b. \tag{5.14}$$

由余项公式 (5.14) 可知, 科茨公式的代数精度为 5.

例 5.4 利用科茨公式计算

$$I = \int_0^1 \frac{4}{1+x^2}\mathrm{d}x$$

的近似值.

解 由科茨公式 (5.13), 有

$$\begin{aligned}\int_0^1 \frac{4}{1+x^2}\mathrm{d}x &\approx \frac{1}{90}[7f(0)+32f(0.25)+12f(0.5)+32f(0.75)+7f(1)]\\ &= \frac{1}{90}\left(7\times 4+32\times\frac{64}{17}+12\times\frac{16}{5}+32\times\frac{64}{25}+7\times 2\right)\\ &= \frac{1}{90}(28+120.4706+38.4+81.92+14)\\ &= \frac{1}{90}\times 282.7906 = 3.1421.\end{aligned}$$

5.3 复化求积公式

从求积系数公式 (5.6) 可以看到, 随着求积节点的增多 (n 的增大), 有可能导致求积系数出现负数 (当 $n \geqslant 8$ 时, 牛顿–科茨求积系数会出现负数). 另一方面, 从求积公式的余项公式也可以看到, 被积函数所用的插值多项式次数越高, 对函数的光滑性要求也越高.

在实际应用往往不采用高阶的牛顿–科茨求积公式, 而是将积分区间划分成若干个相等的小区间, 在各小区间上采用低阶的求积公式 (梯形公式或辛普森公式), 然后利用积分的区间可加性, 把各区间上的积分值加起来, 便得到新的求积公式, 这就是复化求积公式的基本思想.

5.3.1 复化梯形公式及通用程序

1. 复化梯形公式及其误差

将积分区间 $[a,b]$ 剖分为 n 等分, 分点为 $x_k = a + kh\,(k = 0, 1, \cdots, n)$, 其中 $h = (b-a)/n$. 在每个小区间 $[x_k, x_{k+1}]$ 上用梯形公式, 则有

$$\begin{aligned}\int_a^b f(x)\mathrm{d}x &= \sum_{k=0}^{n-1}\int_{x_k}^{x_{k+1}} f(x)\mathrm{d}x \\ &= \sum_{k=0}^{n-1}\left\{\frac{x_{k+1}-x_k}{2}[f(x_k)+f(x_{k+1})]+R_k[f]\right\} \\ &= \frac{h}{2}\sum_{k=0}^{n-1}[f(x_k)+f(x_{k+1})]+\sum_{k=0}^{n-1}R_k[f].\end{aligned}$$

记

$$T_n = \frac{h}{2}\sum_{k=0}^{n-1}[f(x_k)+f(x_{k+1})] = \frac{h}{2}\left[f(a)+f(b)+2\sum_{k=1}^{n-1}f(x_k)\right]. \tag{5.15}$$

公式 (5.15) 称为复化梯形公式, 下标 n 表示将积分区间 $[a,b]$ 的等分数.

注 5.1 *复化梯形公式具有递推性质. 事实上, 若将区间 $2n$ 等分, 这时节点数为 $2n+1$, 设增加的 n 个分点为 $x_{k+\frac{1}{2}}\,(k=0,1,\cdots,n-1)$, 在每个小区间上再用梯形公式, 有*

$$\begin{aligned}T_{2n} &= \sum_{k=0}^{n-1}\left\{\frac{x_{k+\frac{1}{2}}-x_k}{2}[f(x_k)+f(x_{k+\frac{1}{2}})]\right. \\ &\quad \left.+\frac{x_{k+1}-x_{k+\frac{1}{2}}}{2}[f(x_{k+\frac{1}{2}})+f(x_{k+1})]\right\} \\ &= \frac{h}{4}\sum_{k=0}^{n-1}[f(x_k)+2f(x_{k+\frac{1}{2}})+f(x_{k+1})] \\ &= \frac{h}{4}\sum_{k=0}^{n-1}[f(x_k)+f(x_{k+1})]+\frac{h}{2}\sum_{k=0}^{n-1}f(x_{k+\frac{1}{2}}).\end{aligned}$$

从而有

$$T_{2n} = \frac{1}{2}T_n + \frac{h}{2}\sum_{k=0}^{n-1}f(x_{k+\frac{1}{2}}). \tag{5.16}$$

公式 (5.16) 表明, 区间对分后, 只需计算出新分点的函数值, 而原复化梯形公式的作为一个整体保留, 不需要重复计算原节点的函数值, 便可得出对分后的积分值, 从而减少了计算量.

下面我们来讨论复化梯形公式的误差. 记

$$R[T_n] = \sum_{k=0}^{n-1}R_k[f]$$

为复化梯形公式的余项, 则

$$R[T_n] = \sum_{k=0}^{n-1}\Big[-\frac{(x_{k+1}-x_k)^3}{12}f''(\eta_k)\Big] = -\frac{h^3}{12}\sum_{k=0}^{n-1}f''(\eta_k).$$

假设 $f(x)$ 二次连续可微, 则 $f''(x)$ 在 $[a,b]$ 上必存在最大值 M 和最小值 m, 即

$$m \leqslant f''(\eta_k) \leqslant M \quad \Rightarrow \quad nm \leqslant \sum_{k=0}^{n-1}f''(\eta_k) \leqslant nM.$$

由此得

$$m \leqslant \frac{1}{n}\sum_{k=0}^{n-1}f''(\eta_k) \leqslant M.$$

于是由连续函数的介值定理, 必存在一点 $\xi \in [a,b]$, 使得

$$f''(\xi) = \frac{1}{n}\sum_{k=0}^{n-1}f''(\eta_k).$$

从而, 有

$$R[T_n] = -\frac{h^3}{12}nf''(\xi) = -\frac{b-a}{12}h^2 f''(\xi). \tag{5.17}$$

由复化梯形公式的余项公式 (5.17) 可知, 在给定的精度要求下, 可以决定积分区间的等分数 n.

例 5.5 利用复化梯形公式计算积分

$$I = \int_0^1 \frac{\sin x}{x}\mathrm{d}x,$$

使其误差界为 10^{-4}, 应将积分区间 $[0,1]$ 多少等分?

解 设

$$f(x) = \frac{\sin x}{x} = \int_0^1 \cos(tx)\mathrm{d}t,$$

则

$$f^{(k)}(x) = \int_0^1 \frac{\mathrm{d}^k}{\mathrm{d}x^k}[\cos(tx)]\mathrm{d}t = \int_0^1 t^k \cos\big(tx + \frac{k\pi}{2}\big)\mathrm{d}t.$$

从而

$$|f^{(k)}(x)| \leqslant \int_0^1 \big|t^k \cos\big(tx+\frac{k\pi}{2}\big)\big|\mathrm{d}t \leqslant \int_0^1 t^k\mathrm{d}t = \frac{1}{k+1}.$$

故由

$$|R[T_n]| = \Big|-\frac{1-0}{12}h^2 f''(\xi)\Big| \leqslant \frac{h^2}{12}\times\frac{1}{2+1} = \frac{h^2}{36} \leqslant 10^{-4},$$

得 $h \leqslant 6\times 10^{-2}$, 即

$$n = \frac{1}{h} \geqslant \frac{1}{6}\times 10^2 \approx 16.67,$$

所以区间 $[0,1]$ 应该 17 等分才能满足精度要求.

2. 复化梯形公式的通用程序

下面给出复化梯形公式的 MATLAB 通用程序:

• 复化梯形公式 MATLAB 程序

```
%matrap.m
function s=matrap(fun,a,b,n)
%用途: 用复化梯形公式求积分.
%格式: s=matrap(fun,a,b,n)  fun为被积函数, a,b为积分区
%间的左右端点, n为区间的等分数, s返回数值积分值
h=(b-a)/n;
s=0;
for k=1:n-1
   x=a+h*k;
   s=s+feval(fun,x);
end
s=h/2*(feval(fun,a)+feval(fun,b)+2*s);
```

例 5.6 取 $n=10$, 利用复化梯形公式计算积分

$$I=\int_0^1\frac{4}{1+x^2}\mathrm{d}x.$$

解 在 MATLAB 命令窗口执行

```
>> format long
>> fun=inline('4./(1+x.^2)');
>> matrap(fun,0,1,10)
```

计算结果为

```
ans =
     3.13992598890716
```

5.3.2 复化辛普森公式及通用程序

1. 复化辛普森公式及其误差

将积分区间 $[a,b]$ 分成 n 等分, 分点为 $x_k=a+kh\,(k=0,1,\cdots,n)$, 其中 $h=(b-a)/n$. 记区间 $[x_k,x_{k+1}]$ 的中点为 $x_{k+\frac{1}{2}}$, 在每个小区间 $[x_k,x_{k+1}]$ 上用辛普森公式, 则得到所谓的复化辛普森公式:

$$S_n=\sum_{k=0}^{n-1}\frac{x_{k+1}-x_k}{6}\left[f(x_k)+4f(x_{k+\frac{1}{2}})+f(x_{k+1})\right],$$

即

$$S_n = \frac{h}{6}\Big[f(a) + f(b) + 2\sum_{k=1}^{n-1} f(x_k) + 4\sum_{k=0}^{n-1} f(x_{k+\frac{1}{2}})\Big]. \tag{5.18}$$

类似于复化梯形公式, 复化辛普森公式的余项为

$$R[S_n] = -\frac{b-a}{180}\Big(\frac{h}{2}\Big)^4 f^{(4)}(\xi), \quad \xi \in [a,b]. \tag{5.19}$$

事实上

$$\begin{aligned} R[S_n] &= \sum_{k=0}^{n-1} R_k[f] = \sum_{k=0}^{n-1}\Big[-\frac{1}{90}\Big(\frac{x_{k+1}-x_k}{2}\Big)^5 f^{(4)}(\eta_k)\Big] \\ &= -\frac{1}{90}\Big(\frac{h}{2}\Big)^5 \sum_{k=0}^{n-1} f^{(4)}(\eta_k) = -\frac{1}{90}\Big(\frac{h}{2}\Big)^5 n f^{(4)}(\xi) \\ &= -\frac{b-a}{180}\Big(\frac{h}{2}\Big)^4 f^{(4)}(\xi). \end{aligned}$$

例 5.7 利用复化辛普森公式计算积分

$$I = \int_0^1 \frac{\sin x}{x}\mathrm{d}x,$$

使其误差界为 10^{-4}, 应将积分区间 $[0,1]$ 多少等分?

解 利用例 5.5 的结果知

$$|f^{(k)}| \leqslant \frac{1}{k+1}.$$

故由

$$|R[S_n]| \leqslant \Big|-\frac{1-0}{180}\Big(\frac{h}{2}\Big)^4 f^{(4)}(\xi)\Big| \leqslant \frac{h^4}{2880} \times \frac{1}{4+1} = \frac{h^4}{14400} \leqslant 10^{-4},$$

得

$$h \leqslant \frac{1}{5}\sqrt{30},$$

于是

$$n = \frac{1}{h} \geqslant \frac{5}{\sqrt{30}} \approx 0.9129.$$

故取 $n=1$ 即可, 这意味着直接对区间 $[0,1]$ 使用辛普森公式即可达到所要求的精度.

2. 复化辛普森公式的通用程序

下面给出复化辛普森公式的 MATLAB 通用程序：

- 复化辛普森公式 MATLAB 程序

```
%masimp.m
function s=masimp(fun,a,b,n)
%用途: 用复化辛普森公式求积分.
%格式: s=masimp(fun,a,b,n)  fun为被积函数, a,b为积分
%区间的左右端点, n为区间的等分数, s返回数值积分值
h=(b-a)/n; s1=0; s2=0;
for k=1:(n-1)
   x=a+h*k;
   s1=s1+feval(fun,x);
end
for k=0:(n-1)
   x=a+h*(k+1/2);
   s2=s2+feval(fun,x);
end
s=h/6*(feval(fun,a)+feval(fun,b)+2*s1+4*s2);
```

例 5.8 取 $n=10$, 利用复化辛普森公式计算积分

$$I=\int_0^1 \frac{4}{1+x^2}\mathrm{d}x$$

的近似值.

解 在 MATLAB 命令窗口执行

```
>> fun=inline('4./(1+x.^2)');
>> masimp(fun,0,1,10)
```

计算结果为:

```
ans =
      3.14159265296979
```

5.4 龙贝格求积公式

5.4.1 算法推导

在数值求积过程中, 步长的选取是一个很困难的问题. 由余项公式确定步长时, 由于涉及高阶导数估计, 实际中很难应用. 实际应用中数值求积主要依靠自动选择

步长的方法. 对于给定的 n, 当由复化梯形公式 T_n 计算不能满足精度要求时, 可进一步考虑对分每个小区间, 计算 T_{2n}. 由前面的讨论知, T_{2n} 与 T_n 之间存在下面的递推关系:

$$T_{2n}=\frac{1}{2}T_n+\frac{h}{2}\sum_{k=0}^{n-1}f(x_{k+\frac{1}{2}}).$$

上式也称为变步长梯形公式. 由此可方便地计算 $T_1, T_2, T_4, \cdots$, 在增加新节点时, 不浪费原先的计算量, 并且可由 $|T_{2n}-T_n|\leqslant\varepsilon$ 控制计算精度.

由复化梯形公式的余项公式 (5.17) 得

$$R[T_{2n}]=-\frac{b-a}{12}\left(\frac{h}{2}\right)^2 f''(\bar{\xi}),$$

如果 $f''(\bar{\xi})\approx f''(\xi)$, 那么有

$$\frac{R[T_{2n}]}{R[T_n]}=\frac{I^*-T_{2n}}{I^*-T_n}\approx\frac{1}{4}.$$

于是, 由上式可以推得

$$I^*\approx\frac{4}{3}T_{2n}-\frac{1}{3}T_n.$$

上式的右端是否比 T_{2n} 的精度更高呢? 通过实际计算得

$$\begin{aligned}
\frac{4}{3}T_{2n}-\frac{1}{3}T_n&=\frac{4}{3}\left[\frac{1}{2}T_n+\frac{h}{2}\sum_{k=0}^{n-1}f(x_{k+\frac{1}{2}})\right]-\frac{1}{3}T_n\\
&=\frac{1}{3}T_n+\frac{2h}{3}\sum_{k=0}^{n-1}f(x_{k+\frac{1}{2}})\\
&=\frac{1}{3}\left\{\frac{h}{2}\left[f(a)+f(b)+2\sum_{k=1}^{n-1}f(x_k)\right]\right\}+\frac{2h}{3}\sum_{k=0}^{n-1}f(x_{k+\frac{1}{2}})\\
&=\frac{h}{6}\left[f(a)+f(b)+2\sum_{k=1}^{n-1}f(x_k)+4\sum_{k=0}^{n-1}f(x_{k+\frac{1}{2}})\right],
\end{aligned}$$

上式的右端恰为复化辛普森公式, 即

$$S_n=\frac{4}{3}T_{2n}-\frac{1}{3}T_n. \tag{5.20}$$

公式 (5.20) 说明, 复化辛普森公式可以简单地由复化梯形公式的线性组合得到, 这种由较低精度的计算结果通过线性组合得到精度较高的计算结果的方法叫做外推法.

下面我们阐述通过适当的线性组合, 把复化梯形公式的近似值组合成更高精度的积分近似值的方法.

用复化梯形公式, 取区间长度为 h, 公式的值 $T(h)$ 与原积分值

$$I^* = \int_a^b f(x)\mathrm{d}x$$

之间存在如下关系:

$$I^* = T(h) + a_2h^2 + a_4h^4 + \cdots, \tag{5.21}$$

其中 $a_2, a_4, \cdots$ 是与 h 无关的常数. 上式表明, 用 $T(h)$ 来近似 I^*, 其截断误差为

$$R[T(h)] = a_2h^2 + a_4h^4 + \cdots = O(h^2).$$

若对分区间, 新区间的长度变为 $h/2$, 将新区间长度代入 (5.21) 得

$$\begin{aligned} I^* &= T\left(\frac{h}{2}\right) + a_2\left(\frac{h}{2}\right)^2 + a_4\left(\frac{h}{2}\right)^4 + \cdots \\ &= T\left(\frac{h}{2}\right) + \frac{1}{4}a_2h^2 + \frac{1}{16}a_4h^4 + \cdots. \end{aligned} \tag{5.22}$$

用 (5.22) 的 4 倍减去 (5.21) 得

$$3I^* = 4T\left(\frac{h}{2}\right) - T(h) + \left(\frac{1}{4} - 1\right)a_4h^4 + \cdots,$$

则

$$I^* = \frac{4T\left(\frac{h}{2}\right) - T(h)}{3} + O(h^4). \tag{5.23}$$

上式消去了 h^2 项. 若记

$$T_0(h) = T(h), \quad T_1(h) = \frac{4T_0\left(\frac{h}{2}\right) - T_0(h)}{3},$$

则用 $T_1(h)$ 作为 I^* 的近似值, 其误差为 $O(h^4)$.

这样, (5.23) 可以写成

$$I^* = T_1(h) + b_4h^4 + b_6h^6 + \cdots. \tag{5.24}$$

同理, 可用 $T_1(h)$ 的线性组合表示 I^*, 即再将区间对分, 消去 h^4 项, 得

$$I^* = \frac{4^2T_1\left(\frac{h}{2}\right) - T_1(h)}{4^2 - 1} + O(h^6) = T_2(h) + O(h^6), \tag{5.25}$$

其中

$$T_2(h) = \frac{4^2T_1\left(\frac{h}{2}\right) - T_1(h)}{4^2 - 1}.$$

这个推导过程继续下去就得到了龙贝格求积公式. 将它完整地写出来就是

$$T_0(h)=T(h),\quad T_j(h)=\frac{4^jT_{j-1}\left(\frac{h}{2}\right)-T_{j-1}(h)}{4^j-1},\quad j=1,2,\cdots. \tag{5.26}$$

下面我们给出龙贝格求积公式的算法步骤:

算法 5.1 (*龙贝格求积算法*)

步 1 输入 a, b 及精度 ε;

步 2 置 $h=b-a$, $T_1^1=\frac{h}{2}(f(a)+f(b))$;

步 3 置 $i=1$, $j=1$, $n=2$, 对分区间 $[a,b]$, 并计算 T_j^{i+1}, T_{j+1}^{i+1}:

$$T_1^{i+1}=\frac{1}{2}T_1^i+\frac{h}{2}\sum_{k=1}^{n}f(x_{k-\frac{1}{2}}),\quad T_{j+1}^{i+1}=\frac{4^jT_j^{i+1}-T_j^i}{4^j-1};$$

步 4 若不满足终止条件, 作循环:

$$i:=i+1,\ h:=h/2,\ n:=2n,$$

计算: $T_1^{i+1}=\frac{1}{2}T_1^i+\frac{h}{2}\sum_{k=1}^{n}f(x_{k-\frac{1}{2}})$,

对 $j=1,\cdots,i$, 计算: $T_{j+1}^{i+1}=\frac{4^jT_j^{i+1}-T_j^i}{4^j-1}$.

在上面的算法中, 终止条件一般取为

$$|T_{i+1}^{i+1}-T_i^i|\leqslant\varepsilon.$$

5.4.2 通用程序

根据上面的算法 5.1 可编制 MATLAB 通用程序如下:

- *龙贝格求积公式 MATLAB 程序*

```
%maromb.m
function s=maromb(fun,a,b,tol)
%用途: 用龙贝格公式求积分
%格式: s=maromb(fun,a,b,tol), fun 是被积函数, a,b是积分
%下、上限,tol是容许误差, s返回积分近似值
if nargin<4,tol=1e-4;end
i=1; j=1; h=b-a;
T(1,1)=h*(feval(fun,a)+feval(fun,b))/2;
T(i+1,j)=T(i,j)/2+sum(feval(fun,a+h/2:h:b-h/2))*h/2;
```

```
T(i+1,j+1)=(4^j*T(i+1,j)-T(i,j))/(4^j-1);
while (abs(T(i+1,i+1)-T(i,i))>tol)
    i=i+1;h=h/2;
    T(i+1,1)=T(i,1)/2+sum(feval(fun,a+h/2:h:b-h/2))*h/2;
    for j=1:i
        T(i+1,j+1)=(4^j*T(i+1,j)-T(i,j))/(4^j-1);
    end
end
T
s=T(i+1,j+1);
```

例 5.9 取 $\varepsilon = 10^{-6}$, 利用龙贝格求积程序 maromb.m 计算积分

$$I = \int_0^1 \frac{4}{1+x^2}\mathrm{d}x$$

的近似值.

解 在 MATLAB 命令窗口执行

```
>> maromb(inline('4./(1+x.^2)'),0,1,1e-6)
```

计算结果为

```
ans =
    3.14159265363824
```

5.5 高斯型求积公式

5.5.1 算法原理

对已知求积公式 (5.2) 可以讨论它的代数精度, 反之也可以按照代数精度要求导出求积公式. 对于求积公式 (5.2), 当求积节点 $x_k\,(k=0,1,\cdots,n)$ 固定时, 公式 (5.2) 有 $n+1$ 个待定参数, 故此时可要求它满足对 $1,x,\cdots,x^n$ "准确" 这样 $n+1$ 个约束条件, 从而使之至少具有 n 次代数精度.

进一步, 可考虑将 $x_k\,(k=0,1,\cdots,n)$ 也视为待定参数, 这样公式 (5.2) 的待定参数就有 $2n+2$ 个, 从而可望公式 (5.2) 的代数精度达到 $2n+1$. 此类高精度的求积公式称为高斯型公式, 而对应的节点 $x_k\,(k=0,1,\cdots,n)$ 称为区间 $[a,b]$ 上的高斯点.

例 5.10 导出一点高斯公式

$$I = \int_a^b f(x)\mathrm{d}x \approx A_0 f(x_0). \tag{5.27}$$

解 由 (5.27), 有 $2\times0+1=1$ 次代数精度, 因此 (5.27) 对 $f(x)=1$ 和 $f(x)=x$ 是准确的. 故有

$$\begin{cases} A_0=b-a \\ x_0A_0=\dfrac{b^2-a^2}{2} \end{cases} \Rightarrow \begin{cases} A_0=b-a \\ x_0=\dfrac{b+a}{2} \end{cases} \Rightarrow I\approx(b-a)f\left(\frac{a+b}{2}\right).$$

因此, $[a,b]$ 上的 1 阶高斯点为 $(a+b)/2$, 恰为区间的中点.

例 5.11 导出两点高斯公式

$$I=\int_a^b f(x)\mathrm{d}x\approx A_0f(x_0)+A_1f(x_1). \tag{5.28}$$

解 由 (5.28), 有 $2\times1+1=3$ 次代数精度, 因此 (5.28) 对 $f(x)=1$, $f(x)=x$, $f(x)=x^2$ 和 $f(x)=x^3$ 是准确的. 由此可得一个四元线性方程组, 求解困难. 简化运算的方法是, 先设 $a=-1$, $b=1$, 此时有

$$\begin{aligned} &A_0+A_1=2, && ⓐ\\ &x_0A_0+x_1A_1=0, && ⓑ\\ &x_0^2A_0+x_1^2A_1=2/3, && ⓒ\\ &x_0^3A_0+x_1^3A_1=0, && ⓓ \end{aligned}$$

当已知 x_0,x_1 时, 上面的方程 ⓑ,ⓓ 是关于 A_0,A_1 的齐次线性方程组

$$\begin{cases} x_0A_0+x_1A_1=0, \\ x_0^3A_0+x_1^3A_1=0. \end{cases}$$

由于 A_0,A_1 不全为零, 故由克拉默规则有

$$\begin{vmatrix} x_0 & x_1 \\ x_0^3 & x_1^3 \end{vmatrix}=x_0x_1(x_1^2-x_0^2)=0.$$

又易知 $x_0x_1\neq0$, 故 $x_0^2=x_1^2=t$, 代入方程 ⓐ,ⓒ 得, $t=1/3$, 导出

$$x_0=-\frac{1}{\sqrt3},\quad x_1=\frac{1}{\sqrt3}.$$

由此, 再根据方程 ⓐ,ⓑ 得 $A_0=A_1=1$, 即有

$$\int_{-1}^1 f(x)\mathrm{d}x\approx f\left(-\frac{1}{\sqrt3}\right)+f\left(\frac{1}{\sqrt3}\right). \tag{5.29}$$

在一般情形下, 只需通过线性变换

$$x=\frac{b-a}{2}t+\frac{a+b}{2}$$

将 $[a,b]$ 变为 $[-1,1]$. 事实上,

$$\int_a^b f(x)\mathrm{d}x = \int_{-1}^1 f\Big(\frac{b-a}{2}t + \frac{a+b}{2}\Big)\frac{b-a}{2}\mathrm{d}t$$

$$\approx \frac{b-a}{2}\Big[f\Big(-\frac{b-a}{2\sqrt{3}} + \frac{a+b}{2}\Big) + f\Big(\frac{b-a}{2\sqrt{3}} + \frac{a+b}{2}\Big)\Big].$$

于是得到 3 次代数精度的两点高斯公式

$$I = \int_a^b f(x)\mathrm{d}x \approx \frac{b-a}{2}\Big[f\Big(-\frac{b-a}{2\sqrt{3}} + \frac{a+b}{2}\Big) + f\Big(\frac{b-a}{2\sqrt{3}} + \frac{a+b}{2}\Big)\Big].$$

$[a,b]$ 上的 2 阶高斯点为

$$x_0 = -\frac{b-a}{2\sqrt{3}} + \frac{a+b}{2},\quad x_1 = \frac{b-a}{2\sqrt{3}} + \frac{a+b}{2}.$$

更高阶的高斯公式的直接导出比较困难. 以下不加证明地给出 $[-1,1]$ 上高斯点的一般求解方法.

定理 5.2 区间 $[-1,1]$ 上 n 阶高斯点恰为勒让德多项式

$$P_n(x) = \frac{1}{2^n n!}\frac{\mathrm{d}^n}{\mathrm{d}x^n}[(x^2-1)^n]$$

的根.

例 5.12 当 $n=1$ 时, 由

$$\frac{\mathrm{d}^n}{\mathrm{d}x^n}[(x^2-1)^n] = \frac{\mathrm{d}}{\mathrm{d}x}[(x^2-1)] = 2x,$$

得 $[-1,1]$ 上 1 阶高斯点 $x_0 = 0$ (结果与例 5.10 一致).

当 $n=2$ 时, 由

$$\frac{\mathrm{d}^n}{\mathrm{d}x^n}[(x^2-1)^n] = \frac{\mathrm{d}^2}{\mathrm{d}x^2}[(x^2-1)^2] = 12x^2 - 4,$$

得 $[-1,1]$ 上 2 阶高斯点 $x_0 = -1/\sqrt{3},\ x_1 = 1/\sqrt{3}$ (结果与例 5.11 一致).

当 $n=3$ 时, 由

$$\frac{\mathrm{d}^n}{\mathrm{d}x^n}[(x^2-1)^n] = \frac{\mathrm{d}^3}{\mathrm{d}x^3}[(x^2-1)^3] = 120x^3 - 72x,$$

得 $[-1,1]$ 上 3 阶高斯点

$$x_0 = -\sqrt{\frac{3}{5}},\quad x_1 = 0,\quad x_2 = \sqrt{\frac{3}{5}}.$$

然后再用待定系数法解一线性方程组可得相应的求积系数 $A_0 = A_2 = 5/9$, $A_1 = 8/9$. 于是 $[-1,1]$ 上的三点高斯公式为

$$\int_{-1}^{1} f(x)\mathrm{d}x \approx \frac{5}{9}f\Big(-\sqrt{\frac{3}{5}}\Big) + \frac{8}{9}f(0) + \frac{5}{9}f\Big(-\sqrt{\frac{3}{5}}\Big), \tag{5.30}$$

该公式具有 5 次代数精度.

6 阶以下高斯求积公式的高斯点和求积系数如表 5.1.

表 5.1　高斯求积公式的高斯点和对应的求积系数

n	高斯点	求积系数	代数精度
1	0	2	1
2	±0.577350	1	3
3	0	0.888889	5
	±0.774597	0.555556	
4	±0.861136	0.347855	7
	±0.339981	0.652145	
5	0	0.568889	9
	±0.906180	0.236927	
	±0.538469	0.478629	
6	±0.932470	0.131725	11
	±0.661209	0.360762	
	±0.238619	0.467914	

5.5.2　通用程序

根据高斯积分表 5.1, 我们可以编制 6 阶以下高斯求积公式的 MATLAB 通用程序如下:

- 高斯求积公式 MATLAB 程序

```
%magsint.m
function g=magsint(fname,a,b,n,m)
%用途: 用定步长高斯求积公式求函数的积分
%格式: g=magsint(fname,a,b,n,m)  fname是被积函数,
%a,b分别为积分下上限, n为等分数, m为每段高斯点数
switch m
  case 1
     t=0; A=1;
  case 2
```

```
        t=[-1/sqrt(3), 1/sqrt(3)]; A=[1,1];
    case 3
        t=[-sqrt(0.6), 0.0, sqrt(0.6)]; A=[5/9, 8/9, 5/9];
    case 4
        t=[-0.861136, -0.339981, 0.339981, 0.861136];
        A=[0.347855, 0.652145,  0.652145, 0.347855];
    case 5
        t=[-0.906180, -0.538469, 0.0, 0.538469, 0.906180];
        A=[0.236927, 0.478629, 0.568889, 0.478629, 0.236927];
    case 6
        t=[-0.932470,-0.661209,-0.238619,0.238619,0.661209,0.932470];
        A=[0.171325,0.360762,0.467914,0.467914,0.360762,0.171325];
    otherwise
        error('本程序高斯点数只能取1,2,3,4,5,6!');
end
x=linspace(a,b,n+1);
g=0;
for i=1:n
    g=g+gsint(fname,x(i),x(i+1),A,t);
end
%子函数
function g=gsint(fname,a,b,A,t)
g=(b-a)/2*sum(A.*feval(fname,(b-a)/2*t+(a+b)/2));
```

例 5.13 利用高斯求积公式的通用程序 magsint.m 计算积分

$$I=\int_0^1 \frac{4}{1+x^2}\mathrm{d}x \quad 和 \quad I=\int_0^1 \frac{\sin x}{x}\mathrm{d}x$$

的近似值.

解 在 MATLAB 命令窗口执行

```
>> magsint(inline('4./(1+x.^2)'),0,1,2,3)
ans =
     3.14159122238283
>> magsint(inline('4./(1+x.^2)'),0,1,4,4)
ans =
     3.14159265529323
```

```
>> magsint(inline('sin(x)./x'),eps,1,2,3)
ans =
      0.94608307134303
>> magsint(inline('sin(x)./x'),eps,1,4,4)
ans =
      0.94608307062383
```

5.6 数值微分法

5.6.1 差商法

由导数定义可以得到一些简单的数值微分公式.

已知 $f(x)$ 在 $x=a$ 处的导数为

$$f'(a)=\lim_{\Delta x\to 0}\frac{f(a+\Delta x)-f(a)}{\Delta x}.$$

在上式中分别取 $\Delta x=h$ 及 $\Delta x=-h$ 得

$$f'(a)\approx\frac{f(a+h)-f(a)}{h}, \tag{5.31}$$

$$f'(a)\approx\frac{f(a)-f(a-h)}{h}. \tag{5.32}$$

公式 (5.31) 和 (5.32) 分别称为向前差商公式和向后差商公式. 将 (5.31) 和 (5.32) 平均得

$$f'(a)\approx\frac{f(a+h)-f(a-h)}{2h}. \tag{5.33}$$

公式 (5.33) 称为中心差商公式. 这三个公式的几何意义都表示用割线的斜率近似代替切线的斜率.

5.6.2 插值型求导公式

设 $L_n(x)$ 为 $f(x)$ 关于节点 $x_i\,(i=0,1,\cdots,n)$ 的 n 阶拉格朗日插值多项式, 则 $f(x)\approx L_n(x)$. 考虑由

$$f'(x)\approx L_n'(x) \tag{5.34}$$

得到相应的数值微分公式. 分析 (5.34) 的余项

$$\begin{aligned}R(x)=f'(x)-L_n'(x)=[f(x)-L_n(x)]'&=\left[\frac{f^{(n+1)}(\xi)}{(n+1)!}\omega(x)\right]'\\&=\frac{f^{(n+1)}(\xi)}{(n+1)!}\omega'(x)+\frac{\mathrm{d}}{\mathrm{d}x}\left[\frac{f^{(n+1)}(\xi)}{(n+1)!}\right]\omega(x),\end{aligned}$$

其中 $\omega(x)=(x-x_0)(x-x_1)\cdots(x-x_n)$. 我们发现上述余项中的第二项

$$\frac{\mathrm{d}}{\mathrm{d}x}\Big[\frac{f^{(n+1)}(\xi)}{(n+1)!}\Big]\omega(x)$$

一般无法控制 (注意 ξ 与 x 有关). 但是当 x 为某节点时 $\omega(x)=0$, 这时 (5.34) 的余项

$$R(x)=\frac{f^{(n+1)}(\xi)}{(n+1)!}\omega'(x) \tag{5.35}$$

可控制. 因此, 可由 (5.34) 导出节点 x_i 处的数值微分公式:

$$f'(x_i)\approx L_n'(x_i),\quad i=0,1,\cdots,n.$$

例 5.14 利用线性插值分别导出向前差商公式 (5.31) 和向后差商公式 (5.32).

解 取 $x_0=a, x_1=a+h$, 得

$$\begin{aligned}L_1(x)&=\frac{x-x_1}{x_0-x_1}f(x_0)+\frac{x-x_0}{x_1-x_0}f(x_1)\\&=\frac{x-a-h}{-h}f(a)+\frac{x-a}{h}f(a+h),\end{aligned}$$

从而有

$$f'(a)\approx L_1'(a)=\frac{f(a+h)-f(a)}{h}.$$

上式恰为向前差商公式 (5.31).

下面来讨论向前差商公式 (5.31) 的余项. 由于

$$\omega(x)=(x-a)(x-a-h)=(x-a)^2-h(x-a),$$

故

$$\omega'(x)=2(x-a)-h,$$

因此, (5.31) 的余项为

$$f'(a)-L_1'(a)=\frac{f''(\xi)}{2}\omega'(a)=-\frac{h}{2}f''(\xi).$$

同样, 取 $x_0=a-h,\ x_1=a$, 由线性插值可导出向后差商公式 (5.32) 及其余项.

例 5.15 利用抛物插值导出中心差商公式 (5.33) 和 2 阶导数公式.

解 取 $x_0=a-h,\ x_1=a,\ x_2=a+h$, 得

$$\begin{aligned}L_2(x)=&\frac{(x-a)[(x-(a+h)]}{[(a-h)-a][(a-h)-(a+h)]}f(a-h)\\&+\frac{[(x-(a-h)][(x-(a+h)]}{[a-(a-h)][a-(a+h)]}f(a)\end{aligned}$$

$$+\frac{[(x-(a-h)](x-a)}{[(a+h)-(a-h)][(a+h)-a]}f(a+h)$$

$$=\frac{(x-a)^2-h(x-a)}{2h^2}f(a-h)+\frac{(x-a)^2-h^2}{-h^2}f(a)$$

$$+\frac{(x-a)^2+h(x-a)}{2h^2}f(a+h).$$

故

$$L_2'(x)=\frac{2(x-a)-h}{2h^2}f(a-h)+\frac{2(x-a)}{-h^2}f(a)+\frac{2(x-a)+h}{2h^2}f(a+h).$$

从而有

$$f'(a)\approx L_2'(a)=\frac{-h}{2h^2}f(a-h)+\frac{h}{2h^2}f(a+h)=\frac{f(a+h)-f(a-h)}{2h}.$$

下面来讨论中心差商公式 (5.33) 的余项. 由于

$$\omega(x)=(x-a+h)(x-a)(x-a-h)=(x-a)^3-h^2(x-a),$$

故

$$\omega'(x)=3(x-a)^2-h^2.$$

于是中心差商公式 (5.33) 的余项为

$$f'(a)-L_2'(a)=\frac{f^{(3)}(\xi)}{6}\omega'(a)=-\frac{h^2}{6}f^{(3)}(\xi).$$

进一步, 再对 $L_2'(x)$ 求导数得

$$L_2''(x)=\frac{2}{2h^2}f(a-h)+\frac{2}{-h^2}f(a)+\frac{2}{2h^2}f(a+h),$$

从而

$$f''(a)\approx L_2''(a)=\frac{f(a-h)-2f(a)+f(a+h)}{h^2}. \tag{5.36}$$

例 5.16 已知函数 $f(x)=\mathrm{e}^x$ 的数据表:

x	2.6	2.7	2.8
$f(x)$	13.4637	14.8797	16.4446

用二点、三点微分公式计算 $f(x)$ 在 $x=2.7$ 处的一阶、二阶导数的近似值.

解 (1) 由向前差商公式 (5.31) 得

$$f'(2.7)\approx\frac{f(2.8)-f(2.7)}{2.8-2.7}=\frac{16.4446-14.8797}{0.1}=15.649.$$

(2) 由向后差商公式 (5.32) 得

$$f'(2.7) \approx \frac{f(2.7) - f(2.6)}{2.7 - 2.6} = \frac{14.8797 - 13.4637}{0.1} = 14.16.$$

(3) 由中心差商公式 (5.33) 得

$$f'(2.7) \approx \frac{f(2.8) - f(2.6)}{2.8 - 2.6} = \frac{16.4446 - 13.4637}{0.2} = 14.9045.$$

(4) 由公式 (5.36) 得

$$\begin{aligned} f''(2.7) &\approx \frac{f(2.8) - 2f(2.7) + f(2.6)}{0.1^2} \\ &= \frac{16.4446 - 2 \times 14.8797 + 13.4637}{0.01} \\ &= 14.89. \end{aligned}$$

而其准确值

$$f'(2.7) = f''(2.7) = \mathrm{e}^{2.7} = 14.8707,$$

由此可见, 三点公式较两点公式精确.

习　题　5

(Ⅰ) 理论分析题

5.1　证明求积公式

$$\int_{x_0}^{x_1} f(x)\mathrm{d}x \approx \frac{h}{2}[f(x_0) + f(x_1)] - \frac{h^2}{12}[f'(x_1) - f'(x_0)]$$

具有 3 阶代数精度, 其中 $h = x_1 - x_0$.

5.2　证明求积公式

$$\int_{-1}^{1} f(x)\mathrm{d}x \approx \frac{1}{9}[5f(\sqrt{0.6}) + 8f(0) + 5f(-\sqrt{0.6})]$$

对于次数不高于 5 的多项式准确成立, 并计算积分 $\int_0^1 \frac{\sin x}{1+x}\mathrm{d}x$.

5.3　用梯形公式计算积分

$$I = \int_0^1 x^2 \mathrm{d}x$$

的近似值, 并估计截断误差.

5.4　用辛普森公式计算积分

$$I = \int_0^1 x^4 \mathrm{d}x$$

的近似值, 并估计截断误差.

5.5 分别用 4 段梯形公式和 2 段辛普森公式计算下列定积分的近似值, 计算时取 6 位有效数字.

(1) $\int_1^5 \frac{1}{\sqrt{x}}\mathrm{d}x$,　　(2) $\int_1^5 \frac{x}{1+x^2}\mathrm{d}x$.

5.6 考虑用复化梯形公式计算

$$I=\int_0^1 \mathrm{e}^{-x^2}\mathrm{d}x,$$

要使误差小于 0.5×10^{-4}, 那么求积区间 $[0,1]$ 应分多少个子区间? 以此计算积分近似值.

5.7 利用积分 $\int_0^2 \frac{1}{2\sqrt{x}}\mathrm{d}x=\sqrt{2}$ 计算 $\sqrt{2}$ 时, 若采用复化辛普森公式, 问应取多少个节点才能使其误差绝对值不超过 $\frac{1}{2}\times10^{-5}$?

5.8 设 $x_0=0.25,\ x_1=0.5,\ x_2=0.75$.

(1) 推导以 x_0,x_1,x_2 为求积节点在 $[0,1]$ 上的插值型求积公式;

(2) 指出求积公式的代数精度;

(3) 用所求公式计算积分 $I=\int_0^1 x^2\mathrm{d}x$ 的近似值, 并估计截断误差.

5.9 推导下列三种矩形求积公式:

(1) $\int_a^b f(x)\mathrm{d}x=(b-a)f(a)+\frac{f'(\eta)}{2}(b-a)^2$;

(2) $\int_a^b f(x)\mathrm{d}x=(b-a)f(b)-\frac{f'(\eta)}{2}(b-a)^2$;

(3) $\int_a^b f(x)\mathrm{d}x=(b-a)f\Big(\frac{a+b}{2}\Big)+\frac{f''(\eta)}{24}(b-a)^3$.

5.10 给定定积分

$$I=\int_0^1 \frac{\sin x}{x}\mathrm{d}x.$$

(1) 利用复化梯形公式计算上述积分值, 使其截断误差不超过 $\frac{1}{2}\times10^{-3}$.

(2) 取同样的求积节点, 改用复化辛普森公式时, 截断误差是多少?

(3) 要求截断误差不超过 10^{-6}, 若用复化辛普森公式, 应取多少个函数值?

5.11 在区间 $[-1,\ 1]$ 上, 取 $x_1=-\lambda,\ x_2=0,\ x_3=\lambda$, 构造插值求积公式, 并求它的代数精度.

5.12 证明不存在 A_k 及 $x_k\,(k=0,1,\cdots,n)$ 使求积公式

$$\int_a^b f(x)\mathrm{d}x\approx\sum_{k=0}^{n}A_kf(x_k)$$

的代数精度超过 $2n+1$ 次.

5.13　已知求积公式

$$\int_{-2}^{2} f(x)\mathrm{d}x \approx \frac{4}{3}[2f(-1) - f(0) + 2f(1)].$$

试用此公式导出计算 $\int_0^4 f(x)\mathrm{d}x$ 的求积公式.

5.14　用两种不同的方法确定 x_1, x_2, A_1, A_2, 使下面的公式成为高斯求积公式：

$$\int_0^1 f(x)\mathrm{d}x \approx A_1 f(x_1) + A_2 f(x_2).$$

5.15　确定下列数值微分公式的系数, 并导出截断误差表达式：

(1) $f'(0) \approx af(-h) + bf(0) + cf(h)$;

(2) $f'(h) \approx af'(0) + b[f(2h) - f(h)]$.

5.16　已知函数 $y = \mathrm{e}^x$ 的函数值如下：

x	2.5	2.6	2.7	2.8	2.9
y	12.1825	13.4637	14.8797	16.4446	18.1741

试用二点、三点微分公式计算 $x = 2.7$ 处的一阶、二阶导数值.

5.17　已知函数 $f(x) = \dfrac{1}{1+x^2}$ 的数据表如下：

x	1.0	1.1	1.2
$f(x)$	0.2500	0.2268	0.2066

试用三点微分公式计算 $f(x)$ 在 $x = 1.1$ 处的一阶、二阶导数值, 并估计误差.

(Ⅱ) 上机实验题

5.1　取 $n = 10$, 编制复化梯形公式的 MATLAB 程序, 求下列各式右端的定积分的值：

(1) $\ln 2 = \int_2^3 \frac{2}{1-x^2}\mathrm{d}x$;　　(2) $\mathrm{e}^2 = \int_1^2 x\mathrm{e}^x\mathrm{d}x$.

5.2　取 $n = 5$, 编制复化辛普森公式的 MATLAB 程序, 求下列各式右端的定积分的值：

(1) $I = \int_0^1 \frac{\sin x}{x}\mathrm{d}x$;　　(2) $I = \int_0^1 x\mathrm{e}^{-x}\mathrm{d}x$.

5.3　利用算法 5.1 (龙贝格求积算法) 编制 MATLAB 程序计算下列定积分的近似值, 精度为 10^{-8}.

(1) $I = \int_0^4 (x - x^2)\mathrm{e}^{-1.5x}\mathrm{d}x$;　　(2) $I = \int_0^{2\pi} \mathrm{e}^{2x}\sin^2 x\mathrm{d}x$.

5.4　取 $n = 4$, 编制复化三点高斯公式的 MATLAB 程序求下列积分的近似值：

(1) $I = \int_0^1 \frac{\mathrm{e}^x}{\sqrt{1-x^2}}\mathrm{d}x$;　　(2) $I = \int_0^1 \frac{\tan x}{x^{0.7}}\mathrm{d}x$.

第 6 章　常微分方程的数值解法

在工程计算中的许多实际问题的数学模型可以用常微分方程来描述. 但是除了常系数线性微分方程和少数特殊的微分方程可以用解析方法求解外, 绝大多数常微分方程难以求得其精确解. 因此研究常微分方程的近似解法 (数值解法) 具有十分重要的应用意义.

本章主要讨论一阶常微分方程初值问题

$$\begin{cases} y' = f(x, y), \quad a \leqslant x \leqslant b, \\ y(x_0) = y_0 \end{cases} \tag{6.1}$$

的数值解法. 根据常微分方程解的存在唯一性定理, 在 $f(x, y)$ 满足一定的条件下, 解函数 $y = y(x)$ 是唯一存在的.

取步长 h, 记 $x_n = x_0 + nh\,(n = 1, 2, \cdots)$, 按一定的递推公式依次求得各节点 x_n 上解函数值 $y(x_n)$ 的近似值 y_n, 称 $y_0, y_1, \cdots, y_n, \cdots$ 为初值问题 (6.1) 的数值解.

常微分方程初值问题的数值解法一般分为两大类:

(1) 一步法: 这类方法在计算 y_{n+1} 时只用到 x_{n+1}, x_n 和 y_n, 即前一步的值. 因此在有了初值之后就可以逐步往下计算, 其代表是龙格–库塔 (Runge-Kutta) 方法.

(2) 多步法: 这类方法在计算 y_{n+1} 时除用到 x_{n+1}, x_n 和 y_n 以外, 还要用到 x_{n-p}, $y_{n-p}\,(p = 1, \cdots, k; k > 0)$, 即前面 k 步的值. 其代表是亚当斯 (Adams) 方法.

6.1　欧拉方法及其改进

6.1.1　欧拉格式和隐式欧拉格式

由数值微分的向前差商公式可以解决初值问题 (6.1) 中导数 y' 的数值计算问题:

$$y'(x_n) \approx \frac{y(x_n + h) - y(x_n)}{h} = \frac{y(x_{n+1}) - y(x_n)}{h},$$

由此可得

$$y(x_{n+1}) \approx y(x_n) + hy'(x_n).$$

(6.1) 实际上给出

$$y'(x) = f(x, y(x)) \quad \Rightarrow \quad y'(x_n) = f(x_n, y(x_n)).$$

于是有

$$y(x_{n+1}) \approx y(x_n) + hf(x_n, y(x_n)).$$

再由 $y_n \approx y(x_n)$, $y_{n+1} \approx y(x_{n+1})$ 得

$$y_{n+1} = y_n + hf(x_n, y_n), \quad n = 0, 1, \cdots. \tag{6.2}$$

递推公式 (6.2) 称为欧拉格式. 同样, 由向后差商公式可导出下面的差分格式:

$$y_{n+1} = y_n + hf(x_{n+1}, y_{n+1}), \quad n = 0, 1, \cdots. \tag{6.3}$$

公式 (6.3) 为一关于 y_{n+1} 的非线性方程, 称为隐式欧拉格式. 隐式格式使用不方便, 但它一般比显示格式具有更好的数值稳定性.

例 6.1 *考虑初值问题*

$$\begin{cases} y' = -y + x, \quad 0 \leqslant x \leqslant 0.5, \\ y(0) = 0, \end{cases}$$

其精确解为 $y(x) = \mathrm{e}^{-x} + x - 1$. *试分别用欧拉格式和隐式欧拉格式计算其数值解, 并与精确解进行比较.*

解 本题中的 $f(x, y) = -y + x$, 故欧拉格式为

$$y_{n+1} = y_n + hf(x_n, y_n) = (1-h)y_n + hx_n, \quad n = 0, 1, \cdots, \ y_0 = 0.$$

隐式欧拉格式为

$$y_{n+1} = y_n + hf(x_n, y_{n+1}) = y_n - hy_{n+1} + hx_{n+1}, \quad n = 0, 1, \cdots, \ y_0 = 0,$$

整理得

$$y_{n+1} = \frac{1}{1+h}(y_n + hx_{n+1}), \quad n = 0, 1, \cdots, \ y_0 = 0,$$

取 $h = 0.1$, 计算结果如下表:

x_n	0.0	0.1	0.2	0.3	0.4	0.5
欧拉格式	0	0.0000	0.0100	0.0290	0.0561	0.0905
隐式欧拉格式	0	0.0091	0.0264	0.0513	0.0830	0.1209
精确解	0	0.0048	0.0187	0.0408	0.0703	0.1065

常微分方程数值解的误差分析一般比较困难, 通常只考虑第 $n+1$ 步的所谓"局部"截断误差. 我们首先给出局部截断误差的定义.

定义 6.1 对于求解初值问题 (6.1) 的某差分格式, h 为步长. 假设 $y_1, \cdots, y_n$ 是准确的, 称

$$\varepsilon_{n+1} = y(x_{n+1}) - y_{n+1} \tag{6.4}$$

为该差分格式的局部截断误差. 当 $\varepsilon_{n+1} = O(h^{p+1})$ 时, 称该差分格式具有 p 阶精度.

例 6.2 讨论欧拉格式 (6.2) 和隐式欧拉格式 (6.3) 的精度.

解 将 $y(x)$ 在 x_n 处泰勒展开得

$$y(x) = y(x_n) + y'(x_n)(x-x_n) + \frac{y''(x_n)}{2!}(x-x_n)^2 + \frac{y'''(x_n)}{3!}(x-x_n)^3 + \cdots,$$

即有

$$y(x_{n+1}) = y(x_n) + hy'(x_n) + \frac{h^2}{2}y''(x_n) + \frac{h^3}{6}y'''(x_n) + \cdots. \tag{6.5}$$

(1) 对应于欧拉格式 (6.2), 当 $y_n = y(x_n)$ 时,

$$y_{n+1} = y_n + hf(x_n, y_n) = y(x_n) + hf(x_n, y(x_n)) = y(x_n) + hy'(x_n),$$

从而比较 (6.5) 得

$$y(x_{n+1}) - y_{n+1} = O(h^2),$$

即欧拉公式为 1 阶精度.

(2) 对应隐式欧拉公式 (6.3), 由二元函数的泰勒展式

$$\begin{aligned} f(x,y) = & f(x_n, y_n) + f_x(x_n, y_n)(x-x_n) \\ & + f_y(x_n, y_n)(y-y_n) + O[(x-x_n)^2 + (y-y_n)^2], \end{aligned}$$

当 $y_n = y(x_n)$ 时,

$$\begin{aligned} & f(x_{n+1}, y_{n+1}) \\ = & f(x_n, y_n) + hf_x(x_n, y_n) + f_y(x_n, y_n)(y_{n+1} - y_n) + O[h^2 + (y_{n+1} - y_n)^2] \\ = & f(x_n, y(x_n)) + h[f_x(x_n, y(x_n)) + f(x_{n+1}, y_{n+1}) f_y(x_n, y_n) + O(h)] \\ = & y'(x_n) + O(h). \end{aligned}$$

由此得

$$y_{n+1} = y(x_n) + hy'(x_n) + O(h^2).$$

从而比较 (6.5) 式得 $y(x_{n+1}) - y_{n+1} = O(h^2)$, 知隐式欧拉格式也是 1 阶格式.

6.1.2 欧拉格式的改进

对于初值问题 (6.1), 还可以根据导数与积分的关系, 利用数值积分法导出新的求解格式, 可望提高欧拉格式的精度. 事实上, 对 (6.1) 两边在区间 $[x_n, x_{n+1}]$ 上求积分得

$$y(x_{n+1}) = y(x_n) + \int_{x_n}^{x_{n+1}} f(x, y(x))\mathrm{d}x. \tag{6.6}$$

应用数值积分公式求解 (6.6) 中的积分, 可得相应的差分格式. 例如, 由左矩形公式

$$\int_a^b f(x)\mathrm{d}x \approx (b-a)f(a),$$

可导出欧拉公式 (6.2):

$$\begin{aligned} &y(x_{n+1}) \approx y(x_n) + (x_{n+1} - x_n)f(x_n, y(x_n)) \\ &\quad\Rightarrow \quad y_{n+1} = y_n + hf(x_n, y_n). \end{aligned}$$

而由右矩形公式

$$\int_a^b f(x)\mathrm{d}x \approx (b-a)f(b),$$

可导出隐式欧拉公式 (6.3):

$$\begin{aligned} &y(x_{n+1}) \approx y(x_n) + (x_{n+1} - x_n)f(x_{n+1}, y(x_{n+1})) \\ &\quad\Rightarrow \quad y_{n+1} = y_n + hf(x_{n+1}, y_{n+1}). \end{aligned}$$

此外, 由梯形公式

$$\int_a^b f(x)\mathrm{d}x \approx \frac{b-a}{2}[f(a) + f(b)],$$

可导出下面的差分格式

$$y(x_{n+1}) \approx y(x_n) + \frac{h}{2}[f(x_n, y(x_n)) + f(x_{n+1}, y(x_{n+1}))],$$

即

$$y_{n+1} = y_n + \frac{h}{2}[f(x_n, y_n) + f(x_{n+1}, y_{n+1})]. \tag{6.7}$$

公式 (6.7) 称为梯形格式. 同隐式欧拉格式的局部截断误差的推导过程相类似, 可以证明, 对于梯形格式 (6.7) 的局部截断误差为

$$y(x_{n+1}) - y_{n+1} = O(h^3),$$

即梯形格式具有 2 阶精度.

但梯形格式不便于使用, 也是一个隐格式. 为此, 可以考虑用其它的显格式对 (6.7) 式右端的 y_{n+1} 进行预报, 再用 (6.7) 式求解, 这种方法称之为"预报-校正"法.

如先用欧拉格式 (6.2) 对 y_{n+1} 进行计算, 并将结果记为 $\bar{y}_{n+1}$, 再代入 (6.7) 可得"预报-校正"形式的差分格式:

$$\begin{cases} \bar{y}_{n+1} = y_n + hf(x_n, y_n), \\ y_{n+1} = y_n + \dfrac{h}{2}\left[f(x_n, y_n) + f(x_{n+1}, \bar{y}_{n+1})\right]. \end{cases} \tag{6.8}$$

公式 (6.8) 称为改进欧拉格式.

对于改进欧拉格式, 也可以证明其精度是 2 阶的. 事实上, 当 $y_n = y(x_n)$ 时, 由二元函数的泰勒展式

$$\begin{aligned} & f(x_{n+1}, \bar{y}_{n+1}) = f(x_n + h, y_n + hf(x_n, y_n)) \\ &= f(x_n, y_n) + hf_x(x_n, y_n) + hf(x_n, y_n)f_y(x_n, y_n) + O(h^2) \\ &= f(x_n, y(x_n)) + hf_x(x_n, y(x_n)) + hf(x_n, y(x_n))f_y(x_n, y(x_n)) + O(h^2). \end{aligned}$$

注意到

$$y'(x_n) = f(x_n, y(x_n)), \quad y''(x_n) = f_x(x_n, y(x_n)) + y'(x_n)f_y(x_n, y(x_n)),$$

于是有

$$f(x_{n+1}, \bar{y}_{n+1}) = y'(x_n) + hy''(x_n) + O(h^2),$$

代入 (6.8) 的第二式得

$$\begin{aligned} y_{n+1} &= y(x_n) + \frac{h}{2}\left[y'(x_n) + y'(x_n) + hy''(x_n) + O(h^2)\right] \\ &= y(x_n) + hy'(x_n) + \frac{1}{2}h^2y''(x_n) + O(h^3), \end{aligned}$$

从而比较 (6.5) 式得 $y(x_{n+1}) - y_{n+1} = O(h^3)$, 即改进欧拉格式的精度是 2 阶的.

6.1.3 改进欧拉格式通用程序

下面, 我们给出改进欧拉格式的 MATLAB 通用程序.

- 改进欧拉格式 MATLAB 程序

```
%maeuler.m
function [x, y]=maeuler(dyfun,xspan,y0,h)
%用途: 改进欧拉格式解常微分方程y'=f(x,y), y(x0)=y0
%格式: [x,y]=maeuler(dyfun,xspan,y0,h)  dyfun为函
```

```
%数f(x,y), xspan为求解区间[x0,xn], y0为初值y(x0),
%h为步长, x返回节点, y返回数值解
x=xspan(1):h:xspan(2);  y(1)=y0;
for n=1:(length(x0)-1)
    k1=feval(dyfun,x(n),y(n));
    y(n+1)=y(n)+h*k1;
    k2=feval(dyfun,x(n+1),y(n+1));
    y(n+1)=y(n)+h*(k1+k2)/2;
end
x=x'; y=y';
```

例 6.3 取 $h = 0.1$, 用改进欧拉格式的通用程序 maeuler.m 求解下列初值问题:

$$\begin{cases} y' = x + y, \quad 0 \leqslant x \leqslant 0.5, \\ y(0) = 1, \end{cases}$$

并与精确解 $y(x) = 2e^x - x - 1$ 进行比较.

解 在 MATLAB 命令窗口执行:

```
>> clear;dyfun=inline('x+y');
>> [x,y]=maeuler(dyfun, [0, 0.5], 1, 0.1);  [x';y']
ans =
         0    0.1000    0.2000    0.3000    0.4000    0.5000
    1.0000    1.1100    1.2421    1.3985    1.5818    1.7949
>> y1=2.*exp(x)-x-1;  y1'       %精确解
ans =
    1.0000    1.1103    1.2428    1.3997    1.5836    1.7974
>> (y1-y)'      %误差
ans =
    0    0.0003    0.0008    0.0013    0.0018    0.0025
```

6.2 龙格–库塔格式

6.2.1 龙格–库塔法的基本思想

考虑方程 (6.1). 由拉格朗日中值定理, 存在 $0 < \theta < 1$, 使得

$$\frac{y(x_{n+1}) - y(x_n)}{h} = y'(x_n + \theta h).$$

于是, 由 $y' = f(x, y)$ 得

$$y(x_{n+1}) = y(x_n) + hf(x_n + \theta h, y(x_n + \theta h)). \tag{6.9}$$

记 $K^* = f(x_n + \theta h, y(x_n + \theta h))$, 则称 K^* 为区间 $[x_n, x_{n+1}]$ 上的平均斜率. 下面介绍一种由 (6.9) 导出的平均斜率算法, 即所谓的龙格–库塔法.

在欧拉公式中, 简单地取点 x_n 的斜率 $K_1 = f(x_n, y_n)$ 作为平均斜率 K^*, 精度自然很低. 而改进欧拉公式可以写成下列平均化的形式:

$$\begin{cases} y_{n+1} = y_n + h(K_1 + K_2)/2, \\ K_1 = f(x_n, y_n), \\ K_2 = f(x_{n+1}, y_{n+1}). \end{cases} \tag{6.10}$$

上述公式可以理解为: 用 x_n 和 x_{n+1} 两个点的斜率值 K_1 与 K_2 的算术平均值作为平均斜率值 K^*, 而 x_{n+1} 处的斜率值则通过已知信息 y_n 来预测.

如果能够在区间 $[x_n, x_{n+1}]$ 上多预报几个点的斜率值 $K_1, K_2, \cdots, K_r$, 然后取它们的加权平均值

$$\sum_{i=1}^{r} a_i K_i, \quad a_1 + \cdots + a_r = 1$$

作为 K^* 的近似值, 那么计算结果的精度可望更高. 设计区间 $[x_n, x_{n+1}]$ 上 r 个点的预报斜率值 $K_1, K_2, \cdots, K_r$ 及权系数 $a_1, a_2, \cdots, a_r$, 使得差分格式

$$y_{n+1} = y_n + h\sum_{i=1}^{r} a_i K_i \tag{6.11}$$

达到 r 阶精度, 则称公式 (6.11) 为 r 阶龙格–库塔格式.

6.2.2 龙格–库塔格式

考虑差分格式

$$\begin{cases} y_{n+1} = y_n + h(\lambda_1 K_1 + \lambda_2 K_2), \\ K_1 = f(x_n, y_n), \\ K_2 = f(x_n + ph, y_n + phK_1), \quad 0 < p \leqslant 1, \end{cases} \tag{6.12}$$

K_1 视为 $y(x)$ 在点 x_n 处的斜率, K_2 视为 $y(x)$ 在点 $x_{n+p} = x_n + ph$ 处的预报斜率, 若参数 λ_1, λ_2 及 p 的取值使得 (6.12) 具有 2 阶精度, 则称之为二阶龙格–库塔格式.

下面我们来导出二阶龙格–库塔格式 (6.12) 中的参数 λ_1, λ_2 及 p 应满足的条件. 设 $y_n = y(x_n)$ 准确, 得 $K_1 = y'(x_n)$, 由二元函数的泰勒展开得

$$
\begin{aligned}
K_2 &= f(x,y_n) + phf_x(x_n,y_n) + phK_1f_y(x_n,y_n) + O(h^2) \\
&= f(x,y(x_n)) + ph[f_x(x_n,y(x_n)) + y'(x_n)f_y(x_n,y(x_n))] + O(h^2) \\
&= y'(x_n) + phy''(x_n) + O(h^2).
\end{aligned}
$$

于是有

$$
\begin{aligned}
y_{n+1} &= y_n + h[\lambda_1 y'(x_n) + \lambda_2(y'(x_n) + phy''(x_n)) + O(h^2)] \\
&= y_n + (\lambda_1+\lambda_2)hy'(x_n) + \lambda_2 ph^2y''(x_n) + O(h^3).
\end{aligned}
$$

从而由 $y(x_{n+1}) - y_{n+1} = O(h^3)$, 比较 (6.5) 得

$$
\lambda_1 + \lambda_2 = 1, \quad \lambda_2 p = \frac{1}{2}. \tag{6.13}
$$

从 (6.13) 可看出, 三个参数只有两个约束条件, 有一个自由度, 因此二阶龙格–库塔格式是一个系列差分格式. 如取

$$
\lambda_1 = \lambda_2 = \frac{1}{2}, \quad p = 1,
$$

则得到改进欧拉公式 (6.10). 又如取

$$
\lambda = 0, \quad \lambda_2 = 1, \quad p = \frac{1}{2},
$$

则得

$$
\begin{cases}
y_{n+1} = y_n + hK_2, \\
K_1 = f(x_n,y_n), \\
K_2 = f(x_{n+\frac{1}{2}}, y_n + hK_1/2),
\end{cases} \tag{6.14}
$$

其中

$$
x_{n+\frac{1}{2}} = x_n + \frac{1}{2}h.
$$

公式 (6.14) 称为中点格式.

在二阶龙格–库塔格式的基础上可以进一步构造更高阶的龙格–库塔格式. 如对差分格式

$$
\begin{cases}
y_{n+1} = y_n + h(\lambda_1K_1 + \lambda_2K_2 + \lambda_3K_3), \quad \lambda_1+\lambda_2+\lambda_3 = 1, \\
K_1 = f(x_n,y_n), \\
K_2 = f(x_{n+p}, y_n + phK_1), \quad 0 < p \leqslant 1, \\
K_3 = f(x_{n+q}, y_n + qh[(1-\alpha)K_1 + \alpha K_2]), \quad p \leqslant q \leqslant 1,
\end{cases} \tag{6.15}
$$

其中 K_1 视为 $y(x)$ 在点 x_n 处的斜率, K_2, K_3 分别视为 $y(x)$ 在点 $x_{n+ph} = x_n + ph$ 和在点 $x_{n+qh} = x_n + qh$ 处的预报斜率, 若参数 $\lambda_1, \lambda_2, \lambda_3, p, q$ 及 α 的取值使得 (6.15) 具有 3 阶精度, 则称之为三阶龙格-库塔格式.

三阶龙格-库塔格式也不止一个, 最常用的是下面的三阶库塔格式:

$$\begin{cases} y_{n+1} = y_n + \dfrac{h}{6}(K_1 + 4K_2 + K_3), \\ K_1 = f(x_n, y_n), \\ K_2 = f\left(x_{n+\frac{1}{2}}, y_n + \dfrac{h}{2}K_1\right), \\ K_3 = f(x_{n+1}, y_n + h(-K_1 + 2K_2)). \end{cases} \tag{6.16}$$

同样, 最常用的四阶龙格-库塔格式是下面的四阶经典龙格-库塔格式:

$$\begin{cases} y_{n+1} = y_n + \dfrac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4), \\ K_1 = f(x_n, y_n), \\ K_2 = f\left(x_{n+\frac{1}{2}}, y_n + \dfrac{h}{2}K_1\right), \\ K_3 = f\left(x_{n+\frac{1}{2}}, y_n + \dfrac{h}{2}K_2\right), \\ K_4 = f(x_{n+1}, y_n + hK_3). \end{cases} \tag{6.17}$$

例 6.4 取步长 $h = 0.2$, 用四阶龙格-库塔法计算下面的初值问题

$$\begin{cases} y' = y - \dfrac{2x}{y}, \quad 0 \leqslant x \leqslant 1, \\ y(0) = 1, \end{cases}$$

并与精确解比较, 其中精确解为 $y = \sqrt{1+2x}$.

解 由 (6.17) 得

$$y_{n+1} = y_n + \frac{0.2}{6}(K_1 + 2K_2 + 2K_3 + K_4),$$

其中

$$K_1 = y_n - \frac{2x_n}{y_n}, \qquad K_2 = y_n + 0.1K_1 - 2\frac{x_n + 0.1}{y_n + 0.1K_1},$$

$$K_3 = y_n + 0.1K_2 - 2\frac{x_n + 0.1}{y_n + 0.1K_2}, \quad K_4 = y_n + 0.2K_3 - 2\frac{x_n + 0.2}{y_n + 0.2K_3}.$$

计算结果如下表所示:

x_n	y_n	$y(x_n)$
0.0	1.000000	1.000000
0.2	1.183229	1.183216
0.4	1.341667	1.341641
0.6	1.483281	1.483240
0.8	1.612514	1.612452
1.0	1.732142	1.732051

6.2.3 龙格–库塔法的通用程序

我们给出四阶经典龙格–库塔格式 (6.17) 的 MATLAB 通用程序如下.

- 四阶经典龙格–库塔格式 MATLAB 程序

```
%marunge4.m
function [x, y]=marunge4(dyfun,xspan,y0,h)
%用途: 4阶经典龙格库塔格式解常微分方程y'=f(x,y), y(x0)=y0
%格式: [x,y]=marunge4(dyfun,xspan,y0,h)   dyfun为函数f(x,y),
%xspan为求解区间[x0,xn], y0为初值,h为步长,x返回节点,y返回数值解
format short;
x=xspan(1):h:xspan(2);  y(1)=y0;
for n=1:(length(x)-1)
    k1=feval(dyfun, x(n), y(n));
    k2=feval(dyfun, x(n)+h/2, y(n)+h/2*k1);
    k3=feval(dyfun, x(n)+h/2, y(n)+h/2*k2);
    k4=feval(dyfun, x(n+1), y(n)+h*k3);
    y(n+1)=y(n)+h*(k1+2*k2+2*k3+k4)/6;
end
x=x'; y=y';
```

例 6.5 取 $h=0.1$, 用四阶经典龙格–库塔格式的通用程序 marunge4.m 求解下列初值问题:

$$\begin{cases} y' = x + y, & 0 \leqslant 0 \leqslant x \leqslant 0.5, \\ y(0) = 1, \end{cases}$$

并与精确解 $y(x) = 2e^x - x - 1$ 进行比较.

解 在 MATLAB 命令窗口执行:

```
>> clear; dyfun=inline('x+y');
>> [x,y]=marunge4(dyfun,[0,0.5],1,0.1);  [x';y']
```

```
ans =
       0    0.1000    0.2000    0.3000    0.4000    0.5000
  1.0000    1.1103    1.2428    1.3997    1.5836    1.7974
>> y1=2.*exp(x)-x-1;  y1'   %精确解
ans =
  1.0000    1.1103    1.2428    1.3997    1.5836    1.7974
>> (y1-y)'   %误差
ans =
  1.0e-005 *
       0    0.0169    0.0375    0.0621    0.0915    0.1264
```

6.3 收敛性与稳定性

现在我们来讨论前述差分格式的收敛性和绝对稳定性问题. 对于差分格式的误差问题需要从两个方面加以考虑. 首先是截断误差问题. 定义 6.1 已经对差分格式的局部截断误差给出了定性描述. 而本节将要对整体截断误差做出定性描述, 即讨论差分格式的收敛性问题. 其次是舍入误差问题, 本节将要对"试验方程"讨论数据偏差是否会被差分格式放大, 即讨论差分格式的绝对稳定性问题.

6.3.1 收敛性分析

我们首先给出差分格式收敛的定义.

定义 6.2 如果对于任意固定的 $x_N = x_0 + Nh$, 当 $N \to \infty$ (同时 $h \to 0$) 时, 数值解 $y_N \to y(x_N)$, 称求解常微分方程初值问题 (6.1) 的差分格式是收敛的.

根据上述定义, 我们看一个例题.

例 6.6 证明欧拉格式对于求解下列方程收敛:

$$\begin{cases} y' = -y, \\ y(0) = 1. \end{cases} \tag{6.18}$$

证 取 $h = \bar{x}/N$, $x_n = nh\,(n = 0, 1, \cdots, N)$, 则有 $\bar{x} = x_N$. 由欧拉格式得

$$y_{n+1} = y_n + hf(x_n, y_n) = (1-h)y_n = (1-h)^{n+1}y_0 = (1-h)^{n+1}.$$

于是

$$y_N(h) = y_N = (1-h)^N = \left[\left(1 - \frac{\bar{x}}{N}\right)^{-\frac{N}{\bar{x}}}\right]^{-\bar{x}} \to \mathrm{e}^{-\bar{x}} \quad (N \to \infty).$$

又容易发现 $y(x) = \mathrm{e}^{-x}$ 是 (6.18) 的解析解, 故得 $y_N(h) \to y(\bar{x})\,(N \to \infty)$. □

更一般地, 我们有下面的收敛性定理:

定理 6.1 *设差分格式*

$$y_{n+1} = y_n + h\varphi(x_n, y_n, h) \tag{6.19}$$

为解初值问题 (6.1) 的 p 阶格式, 即局部截断误差为 $O(h^{p+1})$. 若增量函数 $\varphi(x,y,h)$ 关于 y 满足 Lipschitz 条件, 即存在 $L>0$, 使 $\forall x, y, \bar{y}, h$ 成立

$$|\varphi(x,y,h) - \varphi(x,\bar{y},h)| \leqslant L|y-\bar{y}|, \tag{6.20}$$

则数值解的整体截断误差为 $e_n = y(x_n) - y_n = O(h^p)$.

证 记

$$\bar{y}_{n+1} = y(x_n) + h\varphi(x_n, y(x_n), h),$$

其中, $\bar{y}_{n+1}$ 表示当 y_n 准确时由差分格式 (6.19) 求得的结果. 则由 (6.19) 是 p 阶格式知, 存在常数 $c>0$, 使

$$|y(x_{n+1}) - \bar{y}_{n+1}| \leqslant ch^{p+1}.$$

从而

$$\begin{aligned}
|y(x_{n+1}) - y_{n+1}| &\leqslant |y(x_{n+1}) - \bar{y}_{n+1}| + |\bar{y}_{n+1} - y_{n+1}| \\
&\leqslant ch^{p+1} + \big|[y(x_n) + h\varphi(x_n, y(x_n), h)] - [y_n + h\varphi(x_n, y_n, h)]\big| \\
&\leqslant ch^{p+1} + |y(x_n) - y_n| + hL|y(x_n) - y_n| \\
&\leqslant ch^{p+1} + (1+hL)|y(x_n) - y_n|,
\end{aligned}$$

由 e_n 的定义, 上式即

$$|e_{n+1}| \leqslant ch^{p+1} + (1+hL)|e_n|. \tag{6.21}$$

注意到若递推式 $u_{n+1} \leqslant a + bu_n$ 成立, 则

$$\begin{aligned}
u_n &\leqslant a + bu_{n-1} \leqslant a + b(a + bu_{n-2}) \leqslant \cdots \\
&\leqslant a(1 + b + b^2 + \cdots + b^{n-1}) + b^n u_0 \\
&= a\frac{1-b^n}{1-b} + b^n u_0.
\end{aligned}$$

那么由 (6.21) 可推得

$$|e_n| \leqslant ch^{p+1}\frac{1-(1+hL)^n}{1-(1+hL)} + (1+hL)^n|e_0| = \frac{ch^p}{L}\left[(1+hL)^n - 1\right].$$

再利用不等式 $(1+x)^n \leqslant e^{nx}$ 可得

$$|e_n| \leqslant \frac{ch^p}{L}(e^{nhL}-1) = \frac{ch^p}{L}\left[e^{(x_n-x_0)L}-1\right].$$

上式表明 $e_n = O(h^p)$. □

例 6.7 *若方程 (6.1) 中的函数 $f(x,y)$ 满足 Lipschitz 条件, 即存在 $L>0$ 使得 $\forall x,y,\bar{y}$, 成立*

$$|f(x,y)-f(x,\bar{y})| \leqslant L|y-\bar{y}|.$$

讨论欧拉格式和改进欧拉格式的收敛性问题.

解 对于欧拉格式 $y_{n+1} = y_n + hf(x_n, y_n)$, 对应的增量函数 $\varphi(x,y,h) = f(x,y)$, 故当 $f(x,y)$ 关于 y 满足 Lipschitz 条件时, 由定理 6.1 知 $y(x_n) - y_n = O(h)$, 从而格式收敛.

而对于改进欧拉格式

$$y_{n+1} = y_n + \frac{h}{2}\left[f(x_n,y_n) + f(x_{n+1}, y_n + hf(x_n,y_n))\right],$$

增量函数为

$$\varphi(x,y,h) = \frac{1}{2}\left[f(x,y) + f(x+h, y+hf(x,y))\right],$$

则有

$$\begin{aligned}
&|\varphi(x,y,h)-\varphi(x,\bar{y},h)| \\
\leqslant\ & \frac{1}{2}|f(x,y)-f(x,\bar{y})| + \frac{1}{2}\left|f(x+h,y+hf(x,y)) - f(x+h,\bar{y}+hf(x,\bar{y}))\right| \\
\leqslant\ & \frac{L}{2}|y-\bar{y}| + \frac{L}{2}\left|[y+hf(x,y)]-[\bar{y}+hf(x,\bar{y})]\right| \\
\leqslant\ & \frac{L}{2}|y-\bar{y}| + \frac{L}{2}|y-\bar{y}| + \frac{hL}{2}|f(x,y)-f(x,\bar{y})| \\
\leqslant\ & \left(\frac{L}{2}+\frac{L}{2}+\frac{hL^2}{2}\right)|y-\bar{y}| = \frac{L}{2}(2+hL)|y-\bar{y}|.
\end{aligned}$$

只要取 $h<1$, 即有

$$|\varphi(x,y,h)-\varphi(x,\bar{y},h)| \leqslant \frac{L}{2}(2+L)|y-\bar{y}| = \bar{L}|y-\bar{y}|,$$

即 $\varphi(x,y,h)$ 满足 (6.20), 故由定理 6.1 知 $y(x_n)-y_n = O(h^2)$, 从而改进欧拉格式是收敛的. □

6.3.2 绝对稳定性

差分格式的数值稳定性问题很难作一般性的讨论. 通常人们仅用试验方程

$$y' = \lambda y, \ \ \lambda < 0 \tag{6.22}$$

作讨论. 这是由于当 $\lambda > 0$ 时, 方程 (6.22) 的解不是渐近稳定的, 即任意初始偏差都可能造成解的巨大差异, 是病态问题. 这里, λ 代表了 $f(x, y)$ 对于 y 偏导数的大致取值.

定义 6.3 设由某差分格式求试验方程 (6.22) 的数值解, 若当 y_n 有扰动 (数据误差或舍入误差) ε 时, y_{n+1} 因此产生的偏差不超过 $|\varepsilon|$, 则称该差分格式是绝对稳定的.

例 6.8 对于试验方程 $y' = \lambda y (\lambda < 0)$, 分别讨论当步长 h 在什么范围取值时, 欧拉格式 (6.2) 和隐式欧拉格式 (6.3) 是绝对稳定的.

解 对于欧拉格式, 由试验方程得

$$y_{n+1} = y_n + hf(x_n, y_n) = (1 + h\lambda)y_n.$$

若 y_n 有扰动 ε_n, y_{n+1} 因此产生偏差 ε_{n+1}, 则有

$$y_{n+1} + \varepsilon_{n+1} = (1 + h\lambda)(y_n + \varepsilon_n) \quad \Rightarrow \quad \varepsilon_{n+1} = (1 + h\lambda)\varepsilon_n.$$

从而, 欧拉格式稳定当且仅当

$$|\varepsilon_{n+1}| = |1 + h\lambda| \cdot |\varepsilon_n| \quad \Leftrightarrow \quad |1 + h\lambda| \leqslant 1.$$

由

$$|1 + h\lambda| < 1 \quad \Rightarrow \quad 1 + h\lambda \geqslant -1 \quad \Rightarrow \quad h\lambda \geqslant -2 \quad \Rightarrow \quad h \leqslant -\frac{2}{\lambda}$$

可知, 欧拉格式是“条件稳定”的, 且 $|\lambda|$ 越大, 稳定区域越小.

又对于隐式欧拉格式, 由试验方程得

$$y_{n+1} = y_n + hf(x_{n+1}, y_{n+1}) = y_n + h\lambda y_{n+1} \quad \Rightarrow \quad y_{n+1} = \frac{y_n}{1 - h\lambda}.$$

若 y_n 有扰动 ε_n, y_{n+1} 因此产生偏差 ε_{n+1}, 则有

$$y_{n+1} + \varepsilon_{n+1} = \frac{y_n + \varepsilon_n}{1 - h\lambda} \quad \Rightarrow \quad \varepsilon_{n+1} = \frac{\varepsilon_n}{1 - h\lambda} \quad \Rightarrow \quad |\varepsilon_{n+1}| = \frac{|\varepsilon_n|}{1 - h\lambda} \leqslant |\varepsilon_n|.$$

由此可见, 隐式欧拉格式是“无条件”绝对稳定的.

用类似的方法可以证明, 改进欧拉格式具有与欧拉格式相仿的稳定性, 而梯形格式是绝对稳定的. 一般地, 隐格式比显格式具有更好的稳定性.

6.4 Adams 格式

6.4.1 Adams 格式推导

一步格式在计算时只用到前面一步的近似值 (比如龙格–库塔格式), 这是一步格式的优点. 但正因为如此, 要提高精度, 需要增加中间函数值的计算, 这就加大了计算量. 下面我们介绍多步格式, 它在计算 y_{n+1} 时除了用到 x_n 上的近似值 y_n 外, 还用到 $x_{n-p}\,(p=1,2,\cdots)$ 上的近似值 y_{n-p}.

线性多步格式的典型代表是 Adams 格式, 它直接利用求解节点的斜率值来提高精度. 其中, 将 $y(x)$ 在 $x_n, x_{n-1}, x_{n-2}, \cdots$ 处斜率值的加权平均作为平均斜率值 K^* 的近似值所得到的格式称为显式 Adams 格式; 而将 $y(x)$ 在 $x_{n+1}, x_n, x_{n-1}, \cdots$ 处斜率值的加权平均作为平均斜率值 K^* 的近似值所得到的格式称为隐式 Adams 格式.

为简化讨论, 记

$$f_k = f(x_k, y_k), \quad k = n+1, n, n-1, \cdots.$$

定义 6.4 *若差分格式*

$$y_{n+1} = y_n + h\sum_{k=1}^{r}\lambda_k f_{n-k+1}, \quad \sum_{k=1}^{r}\lambda_k = 1 \tag{6.23}$$

为 r 阶格式, 则称之为 r 阶显式 Adams 格式. 又若差分格式

$$y_{n+1} = y_n + h\sum_{k=1}^{r}\lambda_k f_{n-k+2}, \quad \sum_{k=1}^{r}\lambda_k = 1 \tag{6.24}$$

为 r 阶格式, 则称之为 r 阶隐式 Adams 格式.

例 6.9 分别导出 2 阶显式与隐式 Adams 格式.

解 设 $f_k = y'(x_k)\,(k=1,2,\cdots,n)$.

(1) 由

$$\begin{aligned}
y_{n+1} &= y_n + h[(1-\lambda)f_n + \lambda f_{n-1}] \\
&= y(x_n) + h[(1-\lambda)y'(x_n) + \lambda y'(x_{n-1})] \\
&= y(x_n) + h\{(1-\lambda)y'(x_n) + \lambda[y'(x_n) - hy''(x_n) + O(h^2)]\} \\
&= y(x_n) + hy'(x_n) - \lambda h^2 y''(x_n) + O(h^3)
\end{aligned}$$

及 $y(x_{n+1}) - y_{n+1} = O(h^3)$, 比较 (6.5) 得 $\lambda = -1/2$, 从而有 2 阶显式 Adams 格式:

$$y_{n+1} = y_n + \frac{h}{2}(3f_n - f_{n-1}). \tag{6.25}$$

(2) 为简便计, 分别用 f, f_x, f_y 表示 $f(x_n, y_n), f_x'(x_n, y_n), f_y'(x_n, y_n)$, 由二元函数的泰勒展式

$$\begin{aligned}f_{n+1} &= f(x_{n+1}, y_{n+1}) \\ &= f + hf_x' + f_y' \cdot (y_{n+1} - y_n) + O[h^2 + (y_{n+1} - y_n)^2],\end{aligned}$$

利用

$$y_{n+1} = y_n + h[(1-\lambda)f_n + \lambda f_{n+1}],$$

得

$$f_{n+1} = f + hf_x' + (1-\lambda)hf \cdot f_y' + \lambda h f_{n+1} \cdot f_y' + O(h^2).$$

那么

$$\begin{aligned}f_{n+1} &= \frac{f + hf_x' + (1-\lambda)hf \cdot f_y' + O(h^2)}{1 - \lambda h f_y'} \\ &= [f + hf_x' + (1-\lambda)hf \cdot f_y' + O(h^2)] \cdot [1 + \lambda h f_y' + O(h^2)] \\ &= f + h(f_x' + f \cdot f_y') + O(h^2) \\ &= y'(x_n) + hy''(x_n) + O(h^2).\end{aligned}$$

这样, 由

$$\begin{aligned}y_{n+1} &= y_n + h[(1-\lambda)f_n + \lambda f_{n+1}] \\ &= y(x_n) + h\{(1-\lambda)y'(x_n) + \lambda[y'(x_n) + hy''(x_n) + O(h^2)]\} \\ &= y(x_n) + hy'(x_n) + \lambda h^2 y''(x_n) + O(h^3)\end{aligned}$$

及 $y(x_{n+1}) - y_{n+1} = O(h^3)$, 比较 (6.5) 得 $\lambda = 1/2$, 从而有 2 阶隐式 Adams 格式:

$$y_{n+1} = y_n + \frac{h}{2}(f_n + f_{n+1}), \tag{6.26}$$

恰为梯形格式.

例 6.10 导出 3 阶显式 Adams 格式.

解 设 $f_k = y'(x_k)\,(k = 1, 2, \cdots, n)$, 则

$$\begin{aligned}y_{n+1} &= y_n + h(\lambda_1 f_n + \lambda_2 f_{n-1} + \lambda_3 f_{n-2}) \\ &= y(x_n) + h[\lambda_1 y'(x_n) + \lambda_2 y'(x_{n-1}) + \lambda_3 y'(x_{n-2})] \\ &= y(x_n) + h\{\lambda_1 y'(x_n) + \lambda_2 [y'(x_n) - hy''(x_n) + \frac{h^2}{2} y'''(x_n)] \\ &\quad + \lambda_3 [y'(x_n) - 2hy''(x_n) + 2h^2 y'''(x_n)] + O(h^3)\}.\end{aligned}$$

整理得

$$\begin{aligned}y_{n+1}=&y(x_n)+h(\lambda_1+\lambda_2+\lambda_3)y'(x_n)+h^2(-\lambda_2-2\lambda_3)y''(x_n)\\&+h^3\Big(\frac{\lambda_2}{2}+2\lambda_3\Big)y'''(x_n)+O(h^4).\end{aligned}$$

由上式及 $y(x_{n+1})-y_{n+1}=O(h^4)$, 比较 (6.5) 得

$$\lambda_1+\lambda_2+\lambda_3=1,\ \ -\lambda_2-2\lambda_3=\frac{1}{2},\ \ \frac{\lambda_2}{2}+2\lambda_3=\frac{1}{6}.$$

解得

$$\lambda_1=\frac{23}{12},\ \ \lambda_2=-\frac{16}{12},\ \ \lambda_1=\frac{5}{12}.$$

从而有 3 阶显式 Adams 格式

$$y_{n+1}=y_n+\frac{h}{12}(23f_n-16f_{n-1}+5f_{n-2}). \tag{6.27}$$

例 6.11 用待定系数法导出 4 阶隐式和显式 Adams 格式.

解 (1) 对于隐式 Adams 格式, 设

$$y_{n+1}=y_n+h(\lambda_1f_{n-2}+\lambda_2f_{n-1}+\lambda_3f_n+\lambda_4f_{n+1}).$$

局部截断误差

$$\begin{aligned}R[y]=&y(x_{n+1})-y_{n+1}=y(x_n+h)-y_{n+1}\\=&y(x_n+h)-y(x_n)-h\big[\lambda_1y'(x_n-2h)\\&+\lambda_2y'(x_n-h)+\lambda_3y'(x_n)+\lambda_4y'(x_n+h)\big].\end{aligned}$$

令 $R[x^k]=0\,(k=1\sim4)$ 及 $x_n=0$, 代入上式得

$$\begin{cases}h(\lambda_1+\lambda_2+\lambda_3+\lambda_4-1)=0,\\h^2(4\lambda_1+2\lambda_2-2\lambda_4+1)=0,\\h^3(12\lambda_1+3\lambda_2+3\lambda_4-1)=0,\\h^4(32\lambda_1+4\lambda_2-4\lambda_4+1)=0.\end{cases}$$

解得

$$\lambda_1=\frac{1}{24},\ \ \lambda_2=-\frac{5}{24},\ \ \lambda_3=\frac{19}{24},\ \ \lambda_4=\frac{9}{24}.$$

得 4 阶隐式格式

$$y_{n+1}=y_n+\frac{h}{24}(f_{n-2}-5f_{n-1}+19f_n+9f_{n+1}). \tag{6.28}$$

公式 (6.28) 称为四阶 Adams 内插公式, 它是一个线性三步四阶隐式公式, 应用十分广泛.

(2) 对于显式 Adams 格式, 设

$$y_{n+1} = y_n + h(\lambda_1 f_n + \lambda_2 f_{n-1} + \lambda_3 f_{n-2} + \lambda_4 f_{n-3}).$$

局部截断误差

$$\begin{aligned} R[y] &= y(x_{n+1}) - y_{n+1} = y(x_n + h) - y_{n+1} \\ &= y(x_n + h) - y(x_n) - h\big[\lambda_1 y'(x_n) + \lambda_2 y'(x_n - h) \\ &\quad + \lambda_3 y'(x_n - 2h) + \lambda_4 y'(x_n - 3h)\big]. \end{aligned}$$

令 $R[x^k] = 0\,(k = 1 \sim 4)$ 及 $x_n = 0$, 代入上式得

$$\begin{cases} h(\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 - 1) = 0, \\ h^2(2\lambda_2 + 4\lambda_3 + 6\lambda_4 + 1) = 0, \\ h^3(3\lambda_2 + 12\lambda_3 + 27\lambda_4 - 1) = 0, \\ h^4(4\lambda_2 + 32\lambda_3 + 108\lambda_4 + 1) = 0. \end{cases}$$

解得

$$\lambda_1 = \frac{55}{24},\quad \lambda_2 = -\frac{59}{24},\quad \lambda_3 = \frac{37}{24},\quad \lambda_4 = -\frac{9}{24}.$$

得 4 阶显式格式

$$y_{n+1} = y_n + \frac{h}{24}(55 f_n - 59 f_{n-1} + 37 f_{n-2} - 9 f_{n-3}). \tag{6.29}$$

公式 (6.29) 称为四阶 Adams 外推公式, 它是一个四步四阶显式公式.

实际应用中, 常将四阶 Adams 外推公式 (6.29) 与内插公式 (6.28) 配套使用, 构成预测–校正公式, 即

$$\begin{cases} p_{n+1} = y_n + \dfrac{h}{24}(55 f_n - 59 f_{n-1} + 37 f_{n-2} - 9 f_{n-3}), \\ y_{n+1} = y_n + \dfrac{h}{24}(f_{n-2} - 5 f_{n-1} + 19 f_n + 9 f(x_{n+1}, p_{n+1})). \end{cases} \tag{6.30}$$

注意到, 外插公式 (6.29) 需要 4 个初值, 通常需要借助于其它差分格式 (如龙格–库塔格式) 计算初值才能启动.

6.4.2 四阶 Adams 格式通用程序

下面给出四阶 Adams 预测–校正公式的 MATLAB 通用程序.

- **四阶 Adams 预测–校正公式 MATLAB 程序**

```
%maadams4.m
function [x, y]=maadams4(dyfun,xspan,y0,h)
%用途：4阶Adams预报-校正格式解常微分方程y'=f(x, y), y(x0)=y0
%格式：[x, y]=maadams4(dyfun,xspan,y0,h)  dyfun为函数f(x,y),
%xspan为求解区间[x0,xn],y0为初值,h为步长,x返回节点,y返回数值解
x=xspan(1):h:xspan(2);
[xx,yy]=marunge4(dyfun,[x(1),x(4)],y0,h);
y(1)=yy(1);y(2)=yy(2);y(3)=yy(3);y(4)=yy(4);
for n=4:(length(x)-1)
  p=y(n)+h/24*(55*feval(dyfun,x(n),y(n))...
    -59*feval(dyfun,x(n-1),y(n-1))+37*feval(dyfun,x(n-2),y(n-2))...
    -9*feval(dyfun,x(n-3),y(n-3)));
  y(n+1)=y(n)+h/24*(feval(dyfun,x(n-2),y(n-2))...
    -5*feval(dyfun,x(n-1),y(n-1))+19*feval(dyfun,x(n),y(n))...
    +9*feval(dyfun,x(n+1),p));
end
x=x'; y=y';
```

例 6.12 取 $h=0.1$, 用四阶 Adams 预报-校正公式的通用程序 maadams4.m 求解下列初值问题:

$$\begin{cases} y' = y - \dfrac{2x}{y}, & 0 \leqslant x \leqslant 1, \\ y(0) = 1. \end{cases}$$

并与精确解 $y(x)=\sqrt{1+2x}$ 进行比较.

解 在 MATLAB 命令窗口执行

```
>> clear; dyfun=inline('y-2*x./y');
>> [x,y]=maadams4(dyfun,[0,1],1,0.1);
>> y1=sqrt(1+2*x);
>> [x,y,y1]
ans =
         0    1.0000    1.0000
    0.1000    1.0954    1.0954
    0.2000    1.1832    1.1832
    0.3000    1.2649    1.2649
    0.4000    1.3416    1.3416
    0.5000    1.4142    1.4142
```

```
    0.6000      1.4832      1.4832
    0.7000      1.5492      1.5492
    0.8000      1.6125      1.6125
    0.9000      1.6733      1.6733
    1.0000      1.7321      1.7321
>> (y-y1)'
ans =
1.0e-005 *
      0    0.0417   0.0789    0.1164    0.0571    0.0271
           0.0127   0.0042   -0.0013   -0.0054   -0.0088
```

6.5 一阶微分方程组和高阶微分方程

前面介绍了一阶常微分方程的各种数值方法，这些方法对常微分方程组和高阶常微分方程同样适用. 为了避免书写上的复杂，下面以二阶常微分方程和两个未知函数的方程组为例来叙述这些方法的计算公式，其截断误差和推导过程与一阶的情形完全一样，不再赘述，只列出计算格式.

6.5.1 一阶常微分方程组

考虑方程组

$$\begin{cases} y' = f(x,y,z), \quad y(x_0) = y_0, \\ z' = g(x,y,z), \quad z(x_0) = z_0. \end{cases} \tag{6.31}$$

(1) 欧拉格式

对 $n = 0,1,2,\cdots$，计算

$$\begin{cases} y_{n+1} = y_n + hf(x_n,y_n,z_n), \quad y(x_0) = y_0, \\ z_{n+1} = z_n + hg(x_n,y_n,z_n), \quad z(x_0) = z_0. \end{cases} \tag{6.32}$$

(2) 改进欧拉格式

对 $n = 0,1,2,\cdots$，计算

$$\begin{cases} p_{n+1} = y_n + hf(x_n,y_n,z_n), \\ q_{n+1} = z_n + hg(x_n,y_n,z_n), \\ y_{n+1} = y_n + \dfrac{h}{2}\left[f(x_n,y_n,z_n) + f(x_{n+1},p_{n+1},q_{n+1})\right], \\ z_{n+1} = z_n + \dfrac{h}{2}\left[g(x_n,y_n,z_n) + g(x_{n+1},p_{n+1},q_{n+1})\right], \end{cases} \tag{6.33}$$

其中 $y(x_0)=y_0$, $z(x_0)=z_0$.

(3) 经典四阶龙格-库塔格式

对 $n=0,1,2,\cdots$, 计算

$$\begin{cases} y_{n+1}=y_n+\dfrac{h}{6}(K_1+2K_2+2K_3+K_4), \\ z_{n+1}=z_n+\dfrac{h}{6}(L_1+2L_2+2L_3+L_4), \end{cases} \tag{6.34}$$

其中

$$\begin{cases} K_1=f(x_n,y_n,z_k), \quad L_1=g(x_n,y_n,z_n), \\ K_2=f\left(x_n+\dfrac{h}{2},y_n+\dfrac{K_1}{2},z_n+\dfrac{L_1}{2}\right), \quad L_2=g\left(x_n+\dfrac{h}{2},y_n+\dfrac{K_1}{2},z_n+\dfrac{L_1}{2}\right), \\ K_3=f\left(x_n+\dfrac{h}{2},y_n+\dfrac{K_2}{2},z_n+\dfrac{L_2}{2}\right), \quad L_3=g\left(x_n+\dfrac{h}{2},y_n+\dfrac{K_2}{2},z_n+\dfrac{L_2}{2}\right), \\ K_4=f(x_n+h,y_n+K_3,z_n+L_3), \quad L_4=g(x_n+h,y_n+K_3,z_n+L_3). \end{cases}$$

(4) Adams 外插格式

记 f_{n-k},g_{n-k} 分别表示 $f(x_{n-k},y_{n-k},z_{n-k}),g(x_{n-k},y_{n-k},z_{n-k})$ $(k=0,1,2,3)$. 对 $n=0,1,2,\cdots$, 计算

$$\begin{cases} y_{n+1}=y_n+\dfrac{h}{24}(55f_n-59f_{n-1}+37f_{n-2}-9f_{n-3}), \\ z_{n+1}=z_n+\dfrac{h}{24}(55g_n-59g_{n-1}+37g_{n-2}-9g_{n-3}), \end{cases} \tag{6.35}$$

其中 $y(x_0)=y_0$, $z(x_0)=z_0$.

(5) Adams 预报-校正格式

记 f_{n-k},g_{n-k} 分别表示 $f(x_{n-k},y_{n-k},z_{n-k}),g(x_{n-k},y_{n-k},z_{n-k})$ $(k=0,1,2,3)$. 对 $n=0,1,2,\cdots$, 计算

$$\begin{cases} p_{n+1}=y_n+\dfrac{h}{24}(55f_n-59f_{n-1}+37f_{n-2}-9f_{n-3}), \\ q_{n+1}=z_n+\dfrac{h}{24}(55g_n-59g_{n-1}+37g_{n-2}-9g_{n-3}), \\ y_{n+1}=y_n+\dfrac{h}{24}(f_{n-2}-5f_{n-1}+19f_n+9f(x_{n+1},p_{n+1},q_{n+1})), \\ z_{n+1}=z_n+\dfrac{h}{24}(g_{n-2}-5g_{n-1}+19g_n+9g(x_{n+1},p_{n+1},q_{n+1})), \end{cases} \tag{6.36}$$

其中 $y(x_0)=y_0$, $z(x_0)=z_0$.

下面给出用经典四阶龙格-库塔格式解常微分方程组的 MATLAB 通用程序:

```
%marunge4s.m
function [x,y]=marunge4s(dyfun,xspan,y0,h)
%用途:4阶经典龙格库塔格式解常微分方程组y'=f(x,y), y(x0)=y0
%格式:[x,y]=marunge4s(dyfun,xspan,y0,h)
%dyfun为向量函数f(x,y), xspan为求解区间[x0,xn],
%y0为初值向量, h为步长, x返回节点, y返回数值解向量
x=xspan(1):h:xspan(2);
y=zeros(length(y0),length(x));
y(:,1)=y0(:);
for n=1:(length(x)-1)
    k1=feval(dyfun,x(n),y(:,n));
    k2=feval(dyfun,x(n)+h/2,y(:,n)+h/2*k1);
    k3=feval(dyfun,x(n)+h/2,y(:,n)+h/2*k2);
    k4=feval(dyfun,x(n+1),y(:,n)+h*k3);
    y(:,n+1)=y(:,n)+h*(k1+2*k2+2*k3+k4)/6;
end
x=x';
y=y';
```

例 6.13 取 $h=0.02$, 利用程序 marunge4s.m 求刚性微分方程组

$$\begin{cases} y' = -0.01y - 99.99z, & y(0)=2, \\ z' = -100z, & z(0)=1 \end{cases}$$

的数值解, 其解析解为 $y=\mathrm{e}^{-0.01x}+\mathrm{e}^{-100x}$, $z=\mathrm{e}^{-100x}$.

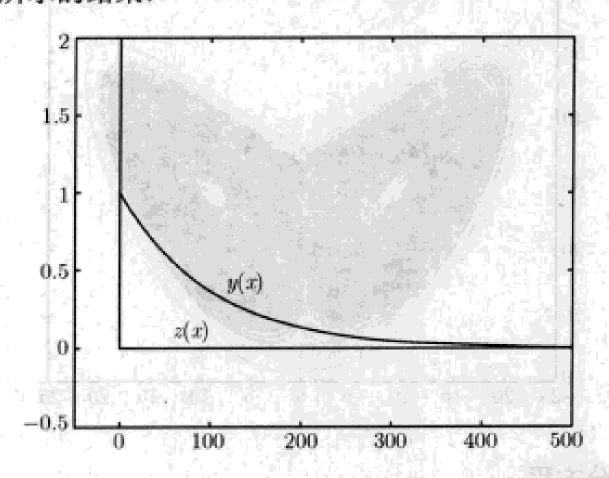
解 首先编写 M 函数 dyfun.m:

```
%dyfun.m
function f=dyfun(t,y)
f(1)=-0.01*y(1)-99.99*y(2);
f(2)=-100*y(2);
f=f(:);
```

再在 MATLAB 命令窗口执行:

```
>> [x,y]=marunge4s(@dyfun,[0 500],[2 1],0.02);
>> plot(x,y);
>> axis([-50 500 -0.5 2]);
>> text(120,0.4,'y(x)');
>> text(70,0.1,'z(x)');
```

得到如下图所示的结果：



例 6.14 考虑下面的 Lorenz 方程组

$$\begin{cases} \dfrac{\mathrm{d}x}{\mathrm{d}t} = -\sigma x + \sigma y, \\ \dfrac{\mathrm{d}y}{\mathrm{d}t} = \alpha x - y - xz, \\ \dfrac{\mathrm{d}z}{\mathrm{d}t} = xy - \beta z. \end{cases}$$

参数 α, β, σ 适当的取值会使系统趋于混沌状态. 取 $\alpha = 30$, $\beta = 2.8$, $\sigma = 12$, 利用经典四阶龙格–库塔法求其数值解, 并绘制 z 随 x 变化的曲线.

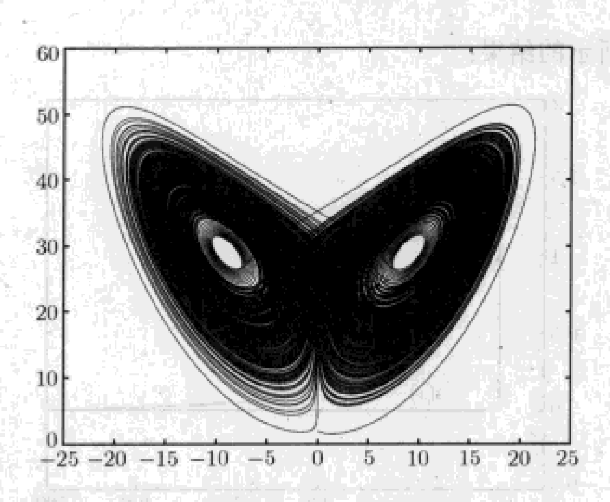
解 首先编写 M 函数 mafun.m：

```
%mafun.m
function ff=mafun(t,y)
b=2.8;r=30;sigma=12;
ff(1)=-sigma*y(1)+sigma*y(2);
ff(2)=r*y(1)-y(2)-y(1)*y(3);
ff(3)=y(1)*y(2)-b*y(3);
ff=ff(:);
```

再在 MATLAB 命令窗口执行：

```
>>  [t,y]=marunge4s(@mafun,[0 500],[0 1 2],0.005);
>>  plot(y(:,1), y(:,3),'r');
```

得到如下图所示的结果：



6.5.2 高阶常微分方程

对于高阶常微分方程, 它总可以化成方程组的形式. 例如, 二阶方程

$$\begin{cases} y'' = g(x, y, y'), \\ y(x_0) = y_0, \quad y'(x_0) = y_0', \end{cases} \tag{6.37}$$

总可以化为一阶方程组

$$\begin{cases} y' = z, \\ z' = g(x, y, z), \\ y(x_0) = y_0, \quad z(x_0) = y_0' = z_0. \end{cases} \tag{6.38}$$

所以没有必要再对高阶方程给出计算公式. 但应注意到, 把高阶方程化为方程组时, 其函数取特定的形式. 因此, 这时的计算公式可以化简. 例如, 对改进欧拉格式, 因 $f(x, y, z) = z$, 故公式可表示为

对 $n = 0, 1, 2, \cdots$, 计算

$$\begin{cases} p_{n+1} = y_n + hz_n, \\ q_{n+1} = z_n + hg(x_n, y_n, z_n), \\ y_{n+1} = y_n + \dfrac{h}{2}(z_n + q_{n+1}), \\ z_{n+1} = z_n + \dfrac{h}{2}\left[g(x_n, y_n, z_n) + g(x_{n+1}, p_{n+1}, q_{n+1})\right], \end{cases} \tag{6.39}$$

其中 $y(x_0) = y_0$, $z(x_0) = y_0' = z_0$.

例 6.15 取 $h = 0.1$, 利用程序 marunge4s.m 求 2 阶方程

$$\begin{cases} y'' = 2y^3, \quad 1 \leqslant x \leqslant 1.5, \\ y(1) = y'(1) = 1 \end{cases}$$

的数值解, 其解析解为

$$y = \frac{1}{x-2}.$$

解　首先将 2 阶方程写成一解方程组的形式

$$\begin{cases} y' = z, \quad y(1) = 1, \\ z' = 2y^3, \quad z(1) = 1. \end{cases}$$

再编写 M 函数 dyfun1.m:

```
%function f=dyfun1(t,y)
f(1)=y(2);
f(2)=2*y(1)^3;
f=f(:);
```

然后再在 MATLAB 命令窗口执行:

```
>> [x,y]=marunge4s(@dyfun1,[1 1.5],[-1 -1],0.1);
>> y1=1./(x-2);  %求精确解
>> [x,y(:,1),y1]
ans =
      1.0000    -1.0000    -1.0000
      1.1000    -1.1111    -1.1111
      1.2000    -1.2500    -1.2500
      1.3000    -1.4285    -1.4286
      1.4000    -1.6666    -1.6667
      1.5000    -1.9998    -2.0000
```

上面的显示结果, 第 1 列是节点, 第 2 列是数值解, 第 3 列是精确解.

习　题　6

(Ⅰ) 理论分析题

6.1　取 $h = 0.1$, 用欧拉公式求解初值问题

$$\begin{cases} y' = -y - xy^2, \quad 0 \leqslant x \leqslant 0.5, \\ y(0) = 1. \end{cases}$$

6.2　取步长 $h = 0.1$, 用隐式欧拉法解初值问题

$$\begin{cases} y' = -2y - 4x, \\ y(0) = 2, \end{cases}$$

其中 $x \in [0, 0.5]$.

6.3　用梯形格式和改进欧拉格式求解初值问题:

$$\begin{cases} y' = x^2 + x - y, \\ y(0) = 0. \end{cases}$$

取步长 $h = 0.1$, 计算到 $x = 0.5$, 并与准确值 $y = \mathrm{e}^{-x} + x^2 - x + 1$ 相比较.

6.4　取 $h = 0.1$, 用改进欧拉公式求解初值问题

$$\begin{cases} y' = \dfrac{1}{2}(x - y), \quad 0 \leqslant x \leqslant 0.5, \\ y(0) = 1. \end{cases}$$

6.5　考虑初值问题:

$$\begin{cases} y' = -y, \\ y(0) = 1. \end{cases}$$

(1) 写出用梯形公式求解上述初值问题的计算格式;

(2) 取步长 $h = 0.1$, 求 $y(0.2)$ 的近似值;

(3) 证明用梯形公式求得的近似解为

$$y_n = \left(\frac{2-h}{2+h}\right)^n,$$

并且当 $h \to 0$ 时, $y_n \to \mathrm{e}^{-x}$.

6.6　取步长 $h = 0.2$, 用四阶经典龙格–库塔法求解下面的初值问题:

$$\begin{cases} y' = 8 - 3y, \quad 1 \leqslant x \leqslant 2, \\ y(1) = 2. \end{cases}$$

6.7　初值问题 $y' = ax + b$, $y(0) = 0$ 的解为

$$y = \frac{1}{2}ax^2 + bx,$$

y_n 是用欧拉法求得的近似解, 证明:

$$y(x_n) - y_n = \frac{1}{2}ahx_n.$$

6.8　证明对任意参数 t, 下列二阶龙格–库塔方法是二阶的:

$$\begin{cases} y_{n+1} = y_n + \dfrac{h}{2}(K_2 + K_3), \\ K_1 = f(x_n, y_n), \\ K_2 = f(x_n + th, y_n + thK_1), \\ K_3 = f(x_n + (1-t)h, y_n + (1-t)hK_1). \end{cases}$$

6.9　设 $f(x, y)$ 满足 Lipschitz 条件, 试证明改进欧拉法和四阶经典龙格–库塔方法都是收敛的.

6.10　试用数值积分公式(中矩形公式)推导求解初值问题 $y'=f(x,y)$, $y(x_0)=y_0$ 的如下中点公式

$$y_{n+1}=y_n+2hf(x_n,y_n)$$

及其局部截断误差

$$T_{n+1}=\frac{1}{3}h^3y'''(\xi).$$

6.11　试用数值积分公式推导求解初值问题 $y'=f(x,y)$, $y(x_0)=y_0$ 的如下公式:

(1) 梯形公式: $y_{n+1}=y_n+\frac{h}{2}(f_n+f_{n+1})$;

(2) 辛普森公式: $y_{n+1}=y_{n-1}+\frac{h}{3}(f_{n-1}+4f_n+f_{n+1})$, 并给出相应公式的局部截断误差.

6.12　对于初值问题

$$y'=-100(y-x^2)+2x,\quad y(0)=1.$$

(1) 用欧拉法求解, 步长 h 取什么范围的值才能使计算稳定?

(2) 若用四阶经典龙格–库塔方法计算, 步长 h 应该如何选取?

(3) 若用梯形公式计算, 步长 h 有无限制?

6.13　证明如下中点公式

$$\begin{cases} y_{n+1}=y_n+hK_2,\\ K_1=f(x_n,y_n),\\ K_2=f\left(x_n+\dfrac{h}{2},y_n+\dfrac{h}{2}K_1\right) \end{cases}$$

是二阶的, 并求其绝对稳定域.

6.14　证明解 $y'=f(x,y)$ 的下列差分公式:

$$y_{n+1}=\frac{1}{2}(y_{n-1}+y_n)+\frac{h}{4}(3y'_{n-1}-y'_n+4y'_{n+1})$$

是二阶的, 并求其局部截断误差.

6.15　已知初值问题

$$\begin{cases} y'=f(x,y),\\ y(x_0)=y_0 \end{cases}$$

的单步数值求解格式

$$y_{n+1}=y_n+\frac{h}{3}[f(x_n,y_n)+2f(x_{n+1},y_{n+1})].$$

求其局部截断误差和阶数, 并证明该方法是是无条件稳定的.

6.16　取 $h=0.1$, 试用欧拉公式求解下面的方程组

$$\begin{cases} y'=3y+2z, & y(0)=0,\\ z'=4y+z, & z(0)=1, \end{cases}\qquad 0\leqslant x\leqslant 0.2.$$

6.17 对二阶常微分方程初值问题

$$\begin{cases} y'' = f(x, y, y'), \quad x_0 \leqslant x \leqslant x_1, \\ y(x_0) = a, \\ y'(x_0) = b, \end{cases}$$

写出用欧拉格式求解的计算公式.

6.18 取 $h = 0.1$, 试用欧拉公式求解下面的二阶常微分方程初值问题

$$\begin{cases} y'' + 4xyy' + 2y^2 = 0, \quad 0 \leqslant x \leqslant 0.2, \\ y(0) = 1, \\ y'(0) = 0. \end{cases}$$

(Ⅱ) 上机实验题

6.1 取 $h = 0.02$, 编制改进欧拉公式的 MATLAB 程序, 求下列常微分方程初值问题的数值解:

$$\begin{cases} y' = -\dfrac{0.9}{1 + 2x}, \quad 0 \leqslant x \leqslant 1, \\ y(0) = 1. \end{cases}$$

6.2 取 $h = 0.1$, 编制四阶经典龙格–库塔公式的 MATLAB 程序, 求下列常微分方程初值问题的数值解:

$$\begin{cases} y' = -\dfrac{2}{y - x}, \quad 0 \leqslant x \leqslant 1, \\ y(0) = 1. \end{cases}$$

6.3 取 $h = 0.1$, 编制四阶 Adams 预报–校正公式的 MATLAB 程序, 求下列常微分方程初值问题的数值解:

$$\begin{cases} y' = \sqrt{x + y}, \quad 0 \leqslant x \leqslant 1, \\ y(0) = 1. \end{cases}$$

6.4 考虑刚性微分方程组:

$$\begin{cases} y' = -1000.25y + 999.75z + 0.5, \quad y(0) = 1, \\ z' = 999.75y - 1000.25z + 0.5, \quad z(0) = -1, \end{cases} \quad 0 \leqslant x \leqslant 50.$$

(1) 编制改进欧拉公式的 MATLAB 程序, 试验得到最小稳定步长;

(2) 编制四阶经典龙格–库塔公式的 MATLAB 程序, 试验得到最小稳定步长.

第 7 章　非线性方程迭代解法

在工程计算和科学研究中, 如电路和电力系统计算、非线性力学、非线性积分和微分方程等多领域都要遇到非线性方程的求根问题. 本章主要讨论求解一元非线性性方程

$$f(x) = 0 \tag{7.1}$$

的数值方法, 其中 $f(x)$ 是连续的非线性函数. 而方程按 $f(x)$ 是多项式或超越函数又分别称为代数方程和超越方程. 例如, 代数方程

$$x^4 - 8x^3 + 26x^2 - 43x + 17 = 0,$$

超越方程

$$\sin\frac{\pi x}{2} - \mathrm{e}^{-x} = 0.$$

已经证明, 对于 5 次及 5 次以上的一元多项式方程不存在精确的求根公式; 至于超越方程就更难求其精确解了. 鉴于此, 如何求得满足一定精度的方程的近似根, 已成为广大科研工作者迫切需要解决的问题.

本章的目的就是介绍求解非线性方程的数值方法. 主要介绍二分法, 迭代法及其加速方法, 牛顿型算法, 并讨论算法的收敛性、收敛速度和计算效率等问题.

7.1　根的搜索与二分法

7.1.1　隔根区间

在用近似方法求方程的根时, 需要知道方程的根所在的区间. 如果在区间 $[a, b]$ 内只有方程 $f(x) = 0$ 的一个根, 则称区间 $[a, b]$ 为隔根区间. 通常可以用逐步扫描法来寻找方程 $f(x) = 0$ 的隔根区间. 算法的基本思想是: 先确定方程 $f(x) = 0$ 的所有实根所在区间 $[a, b]$, 再按选定的步长 $h = (b - a)/n$ (n 为正整数), 逐点计算 $x_k = a + kh$ 处的函数值 $f(x_k)\,(k = 0, 1, \cdots, n)$, 当 $f(x_k)$ 与 $f(x_{k+1})$ 的值异号时, 则 $[x_k, x_{k+1}]$ 即为方程 $f(x) = 0$ 的一个隔根区间.

算法 7.1 (逐步搜索法)

步 1　输入 a, b, h;

步 2　置 $a1 := a,\ b1 := a + h$;

步 3　while $(b1 < b)$ (循环开始)

if $f(a1)\cdot f(b1)<0$ then

$x1(k):=a1,\ x2(k):=b1;$

else

$a1:=b1;\ b1:=b1+h;$

continue; (返回到循环的入口)

end

$a1:=b1;\ b1:=b1+h;$

$k:=k+1;$

end (循环结束)

步 4 输出有根区间 $[x1(k),x2(k)]$.

根据算法 7.1 编制 MATLAB 程序如下：

- 算法 7.1 的 MATLAB 程序

```
%masearch.m
function masearch(fun,a,b,h)
%用途：搜索非线性方程f(x)=0的有根区间
%格式：masearch(fun,a,b,h)  fun为函数表达式，
% a，b为区间左右端点，h为搜索步长
n=(b-a)/h; x1=zeros(1,n); x2=zeros(1,n);
a1=a; b1=a1+h; k=1;
while(b1<b)
  if feval(fun,a1)*feval(fun,b1)<0
    x1(k)=a1; x2(k)=b1;
  else
    a1=b1; b1=a1+h; continue;
  end
  a1=b1; b1=a1+h;  k=k+1;
end
for i=1:k
  if x1(i)-x2(i)~=0  [x1(i),x2(i)]  end
end
```

例 7.1 试用逐步搜索法确定方程

$$f(x)=x^3+x^2-3x-3=0$$

的有根区间.

容易看出，当 $|x|>3$ 时，$f(x)$ 保持符号不变，故其根必定全部落在区间 $[-3,3]$ 内，即可初步确定 $a=-3$，$b=3$. 取步长 $h=0.6$，利用算法 7.1 的通用程序 masearch.m，在 MATLAB 命令窗口执行：

```
>> masearch(inline('x^3+x^2-3*x-3'),-3,3,0.6)
```

得计算结果：

```
ans =
   -1.8000   -1.2000
ans =
   -1.2000   -0.6000
ans =
    1.2000    1.8000
```

即方程有三个根，分别在区间 $[-1.8.-1.2]$，$[-1.2,-0.6]$，$[1.2,1.8]$ 内.

7.1.2 二分法及其程序实现

二分法的基本思想是通过计算隔根区间的中点，逐步将隔根区间缩小，从而可得到方程的近似根数列. 具体地说，设 $f(x)$ 为连续函数，又设方程的隔根区间为 $[a,b]$，即 $f(a)f(b)<0$. 记 $a_0:=a$，$b_0:=b$，取其中点 $x_0=(a_0+b_0)/2$，若 $f(a_0)f(x_0)<0$，则去掉右半区间，即令 $a_1:=a_0$，$b_1:=x_0$；否则，去掉左半区间，即令 $a_1:=x_0$，$b_1=b_0$. 一般地，记当前有根区间为 $[a_k,b_k]$，取

$$x_k=\frac{a_k+b_k}{2}, \tag{7.2}$$

若 $f(a_k)f(x_k)<0$，则令 $a_{k+1}:=a_k$，$b_{k+1}:=x_k$；否则，令 $a_{k+1}:=x_k$，$b_{k+1}:=b_k$；再取 $x_{k+1}=(a_{k+1}+b_{k+1})/2$，一直做下去，直到满足精度为止.

算法 7.2 (二分法)

步 1 由算法 7.1 得到隔根区间 $[a,b]$，设定精度要求 ε；

步 2 置 $x:=(a+b)/2$；

步 3 若 $f(x)=0$，输出 x，停算；否则，转步 4；

步 4 若 $f(a)\cdot f(x)<0$，则置 $b:=x$；否则，置 $a:=x$；

步 5 置 $x=(a+b)/2$，若 $|b-a|<\varepsilon$，输出 x，停算；否则，转步 3.

根据算法 7.2 编制 MATLAB 通用程序如下：

- 二分法 MATLAB 程序

```
%mabisec.m
function x=mabisec(fun,a,b,ep)
%用途: 用二分法求非线性方程f(x)=0有根区间[a,b]中的一个根
%格式: x=mabisec(fun,a,b,ep)  fun为函数表达式,
```

```
% a, b为区间左右端点, ep为精度, x返回近似根
x=(a+b)/2.0;
k=0;
while abs(feval(fun,x))>ep|(b-a>ep)
   if feval(fun,x)*feval(fun,a)<0
      b=x;
   else
      a=x;
   end
   x=(a+b)/2.0;
   k=k+1;
end
disp(['k=',num2str(k)])
```

例 7.2 *用二分法程序* mabisec.m *求方程* $f(x)=xe^x-1=0$ *在* $[0,1]$ *内的一个实根. 取定精度* $\varepsilon=10^{-5}$.

解 在 MATLAB 命令窗口执行:

```
>> x=mabisec(inline('x*exp(x)-1'),0,1,1e-5)
```

得计算结果:

```
k=16
x =
   0.56714630126953
```

即迭代 16 之后, 得到满足给定精度的近似根.

7.1.3 二分法的收敛性分析

现在来估计由 (7.2) 式产生的点列 $\{x_k\}$ 与方程的根 x^* 之间的误差是多少. 根据二分法的基本思想, x_k 与 x^* 的误差不会超过区间 $[a_k,b_k]$ 长度的一半, 并注意到每一个小区间的长度是前一个区间长的一半, 因此有

$$|x_k-x^*|\leqslant\frac{b_k-a_k}{2}=\frac{b_{k-1}-a_{k-1}}{2^2}=\cdots=\frac{b_0-a_0}{2^{k+1}},$$

即

$$|x_k-x^*|\leqslant\frac{1}{2^{k+1}}(b-a). \tag{7.3}$$

由上式可知, 当 $k\to\infty$ 时, 有 $|x_k-x^*|\to 0$, 即

$$\lim_{k\to\infty}x_k=x^*.$$

从以上推导过程可以看到，序列 $\{x_k\}$ 的收敛性与初始区间 $[a,b]$ 无关. 对于任意给定的初始区间 $[a,b]$，序列 $\{x_k\}$ 均是收敛的，因此，二分法是大范围收敛的.

例 7.3 *用二分法求方程 $x^3-3x-1=0$ 在区间 $[1,2]$ 内的根，使其精度达到两位有效数字. 问需要将区间二分多少次？并求出满足精度的近似根.*

解 根据 (7.3) 可以估计二分次数 k 的大小. 设

$$|x_k-x^*|\leqslant\frac{1}{2^{k+1}}(b-a)\leqslant\varepsilon,$$

其中 $a=1$, $b=2$, 精度 $\varepsilon=0.05$, 那么可求得 $k\geqslant(\ln 20/\ln 2)-1\approx 3.3219$, 取 $k=4$ 即可. 用公式 (7.2) 求解得 $x^*\approx x_4=1.9063$, 具体过程见下表：

k	a_k	b_k	x_k	b_k-a_k	$f(a_k)f(x_k)$
0	1	2	1.5	1	+
1	1.5	2	1.75	0.5	+
2	1.75	2	1.875	0.25	+
3	1.875	2	1.9375	0.125	−
4	1.875	1.9375	1.90625	0.0625	

根据上述讨论，二分法具有计算简单、方法可靠并且有大范围收敛性的优点. 缺点是收敛缓慢 (只有线性收敛速度)，并且不能求重根和复根. 通常用二分法为其它的求根方法 (例如牛顿法) 提供较好的初始近似值，再用其它的求根方法精确化.

7.2 简单迭代法及其加速技巧

7.2.1 迭代法的基本思想

能够得到递推形式的算法称为迭代法. 迭代法是数值方法中最常用的一种方法，它是一种逐次逼近的方法. 其基本思想是：先给出方程的根的一个近似值 (初始值)，反复使用某递推公式，校正根的近似值，使之逐渐精确化，最后得到满足精度要求的方程的近似解.

基于上述思想，将方程 $f(x)=0$ 改写成其等价形式

$$x=\varphi(x). \tag{7.4}$$

取方程根的某一近似值 x_0 作为迭代的初始点，由函数 $\varphi(x)$ 可计算出 x_1，即 $x_1=\varphi(x_0)$，如此下去，……，设当前点为 x_k，由 $\varphi(x)$ 计算出 x_{k+1}，即

$$x_{k+1}=\varphi(x_k),\qquad k=0,1,\cdots, \tag{7.5}$$

这里，称 $\varphi(x)$ 为迭代函数，而称式 (7.5) 为迭代公式.

若序列 $\{x_k\}$ 存在极限 x^*, 即

$$\lim_{k\to\infty} x_k = x^*,$$

则称迭代过程 (或迭代公式) 是收敛的. 如果函数 $\varphi(x)$ 连续, 在式 (7.5) 两端取极限得到

$$x^* = \varphi(x^*),$$

则 x^* 是方程 (7.4) 的根. 我们也称 x^* 是函数 $\varphi(x)$ 的一个不动点, 因此也称迭代公式 (7.5) 为不动点迭代. 在迭代公式 (7.5) 中, 由于 x_{k+1} 仅由 x_k 决定, 因此这是一个单步迭代公式.

由上述讨论, 可得到如下算法.

算法 7.3 (简单迭代法)

步 1 取初始点 x_0, 最大迭代次数 N 和精度要求 ε, 置 $k := 0$;

步 2 计算 x_{k+1};

步 3 若 $|x_{k+1} - x_k| < \varepsilon$, 则停算;

步 4 若 $k = N$, 则停算; 否则, 置 $k := k + 1$, 转步 2.

由以上算法过程可以看出, 一旦确定了迭代函数, 算法 7.3 的程序实现非常简单.

根据算法 7.3 编制 MATLAB 通用程序如下:

- 简单迭代法 MATLAB 程序

```
%maiter.m
function x=maiter(phi,x0,ep,N)
%用途: 用简单迭代法求非线性方程f(x)=0有根区间[a,b]中的一个根
%格式: x= maiter(phi,x0,ep,N)  fun为phi(x)的函数句柄, x0为初值,
%ep为精度(默认1e-4), N为最大迭代次数(默认500), x返回近似根
if nargin<4 N=500;end
if nargin<3 ep=1e-4;end
k=0;
while k<N
   x=feval(phi,x0);
   if abs(x-x0)<ep
      break;
   end
   x0=x;k=k+1;
end
```

```
if k==N, warning('已达迭代次数上限'); end
disp(['k=',num2str(k)])
```

例 7.4 *用简单迭代法通用程序 maiter.m 求方程 $f(x) = xe^x - 1 = 0$ 在 $[0,1]$ 内的一个实根. 取定精度 $\varepsilon = 10^{-5}$, 初始点为 $x_0 = 0.5$.*

解 迭代法成功的关键在如何适当选取迭代函数. 本问题可将原方程等价变形为至少下列三种形式

$$x = e^{-x}, \quad x = -\ln x, \quad x = x + xe^x - 1.$$

经过粗略的观察, 可发现后两种等价变形是不可取的 (稍后还有详细论述). 故取迭代函数为 $\varphi(x) = e^{-x}$, 迭代公式为

$$x_{k+1} = e^{-x_k},$$

在 MATLAB 命令窗口执行:

```
>> x=maiter(inline('exp(-x)'),0.5,1e-5)
k=17
x =
   0.56714076326981
```

7.2.2 收敛性和误差分析

用简单迭代法求解非线性方程的关键在于适当地构造迭代公式, 不同的迭代公式收敛的速度不同, 甚至不收敛. 例如, 用迭代法求解方程 $x^3 - x - 1 = 0$, 可以将方程等价变形为至少下面四种形式:

$$x = \sqrt[3]{x+1}, \quad x = x^3 - 1, \quad x = \sqrt{1 + \frac{1}{x}}, \quad x = \frac{x^3 + x - 1}{2}, \cdots$$

由此, 可以得到不同的迭代公式

$$\text{(I)} \quad x_{k+1} = \sqrt[3]{x_k + 1}, \quad \text{(II)} \quad x_{k+1} = x_k^3 - 1,$$

$$\text{(III)} \quad x_{k+1} = \sqrt{1 + \tfrac{1}{x_k}}, \quad \text{(IV)} \quad x_{k+1} = \frac{x_k^3 + x_k - 1}{2}.$$

利用前面的 MATLAB 进行计算, 取初始值 $x_0 = 1.5$, 发现格式 (II) 和 (IV) 是不收敛的, 而格式 (I) 迭代 6 次, 格式 (III) 迭代 8 次即可达到满意的精度 ($< 10^{-5}$).

那么, 现在的问题是, 当迭代公式 (或迭代函数) 满足什么样的条件时, 才能保证所产生的迭代序列收敛 (到方程的解) 呢? 我们有下面的收敛性定理:

定理 7.1 设函数 $\varphi(x)$ 在区间 $[a,b]$ 上有连续的一阶导数, 并满足:

(1) $a \leqslant \varphi(x) \leqslant b, \ \forall x \in [a,b]$; (2) $|\varphi'(x)| \leqslant L < 1, \ \forall x \in [a,b]$.

则 (1) 函数 $\varphi(x)$ 在区间 $[a,b]$ 上存在唯一的不动点 x^*, 即 $x^* = \varphi(x^*)$; (2) 对任何 $x_0 \in [a,b]$, 由迭代公式 (7.5) 得到的迭代序列 $\{x_k\}$ 均收敛到方程的解 x^*.

证 (1) 先证明存在性. 作函数 $g(x) = x - \varphi(x)$, 则由题设 $g(a) = a - \varphi(a) \leqslant 0$, $g(b) = b - \varphi(b) \geqslant 0$, 由根的存在定理, 至少存在一个 x^*, 使得 $g(x^*) = 0$, 即 $x = \varphi(x^*)$.

再证唯一性. 假设存在两个解 x^* 和 $\bar{x}$, 即

$$x^* = \varphi(x^*), \quad \bar{x} = \varphi(\bar{x}), \quad x^*, \ \bar{x} \in [a,b],$$

那么, 由拉格朗日中值定理可得

$$|x^* - \bar{x}| = |\varphi(x^*) - \varphi(\bar{x})| \leqslant |\varphi'(\xi)| \cdot |x^* - \bar{x}| \leqslant L|x^* - \bar{x}|, \tag{7.6}$$

因为 $L < 1$, 式 (7.6) 成立必然有 $x^* = \bar{x}$.

(2) 由拉格朗日中值定理及条件 (2), 可得

$$\begin{aligned} |x_k - x^*| &= |\varphi(x_{k-1}) - \varphi(x^*)| = |\varphi'(\xi_{k-1}) \cdot (x_{k-1} - x^*)| \\ &\leqslant L|x_{k-1} - x^*| \leqslant L^2|x_{k-2} - x^*| \leqslant \cdots \leqslant L^k|x_0 - x^*|. \end{aligned}$$

由于 $L < 1$, 故当 $k \to \infty$ 时, $L^k \to 0$, 从而 $x_k \to x^*(k \to \infty)$. □

注 7.1 由定理 7.1 的证明过程可以看出, 条件 (2) 可以用 Lipschitz 条件来替代, 即: 存在常数 L 且 $0 < L < 1$, 使得

$$|\varphi(x) - \varphi(y)| \leqslant L|x - y|, \quad \forall x, y \in [a,b], \tag{7.7}$$

定理的结论依然成立.

定理 7.1 只是定性地指出了在满足一定的条件下, 迭代序列收敛到方程的解, 并没有定量地给出近似解与真解的误差, 这样, 在构造算法时, 无法确定终止条件. 下面的定理给出了算法 7.1 的误差估计.

定理 7.2 设定理 7.1 的条件成立, 则

$$|x_k - x^*| \leqslant \frac{L^k}{1-L}|x_1 - x_0|, \tag{7.8}$$

$$|x_k - x^*| \leqslant \frac{1}{1-L}|x_{k+1} - x_k|. \tag{7.9}$$

证 由定理 7.1 的证明过程可知,

$$|x_{k+1} - x_k| \leqslant L|x_k - x_{k-1}| \leqslant \cdots \leqslant L^k|x_1 - x_0|. \tag{7.10}$$

反复利用 (7.10) 得到

$$\begin{aligned}|x_{k+p}-x_k| &\leqslant |x_{k+p}-x_{k+p-1}|+|x_{k+p-1}-x_{k+p-2}+\cdots+|x_{k+1}-x_k| \\ &\leqslant (L^{k+p-1}+L^{k+p-2}+\cdots+L^k)|x_1-x_0| \\ &\leqslant \frac{L^k}{1-L}|x_1-x_0|,\end{aligned}$$

在上式中, 令 $p\to\infty$, 即得 (7.8).

用同样的方法, 可以得到

$$\begin{aligned}|x_{k+p}-x_k| &\leqslant |x_{k+p}-x_{k+p-1}|+|x_{k+p-1}-x_{k+p-2}|+\cdots+|x_{k+1}-x_k| \\ &\leqslant (L^{p-1}+L^{p-2}+\cdots+L+1)|x_{k+1}-x_k| \\ &\leqslant \frac{1}{1-L}|x_{k+1}-x_k|,\end{aligned}$$

在上式中, 令 $p\to\infty$, 即得 (7.9). □

上述定理所讨论的收敛性是在整个求解区间 $[a,b]$ 上论述的, 这种收敛性称为全局收敛性. 在实际使用迭代法时, 有时候不方便验证整个区间上的收敛条件, 而实际上只考察不动点 x^* 附近的收敛性, 因此称为局部收敛性.

定义 7.1 设 x^* 是迭代函数 $\varphi(x)$ 的不动点, 如果存在 x^* 的某个邻域 $N(x^*,\delta)=(x^*-\delta,x+\delta)$, 使得对任意的 $x_0\in N(x^*,\delta)$, 由迭代公式 (7.5) 产生的序列 $\{x_k\}\subset N(x^*,\delta)$, 且收敛到 x^*, 则称迭代公式 (7.5) 局部收敛.

定理 7.3 设 x^* 是方程 $x=\varphi(x)$ 的根, $\varphi'(x)$ 在 x^* 的某个邻域内连续且有 $|\varphi'(x^*)|<1$, 则迭代公式 (7.5) 局部收敛.

证 由定理的条件, 存在 $\delta>0$, 使得 $\forall x\in N(x^*,\delta)$ 时, 有 $|\varphi'(x)|\leqslant L<1$. 因此, 我们有

$$|x_{k+1}-x^*|=|\varphi(x_k)-\varphi(x^*)|=|\varphi'(\xi_k)\cdot(x_k-x^*)|\leqslant L|x_k-x^*|<|x_k-x^*|,$$

所以当 $x_k\in N(x^*,\delta)$ 时, 有 $x_{k+1}\in N(x^*,\delta)$. 由定理 7.1 知, 迭代公式 (7.5) 局部收敛. □

例 7.5 利用适当的迭代格式证明

$$1+\cfrac{1}{1+\cfrac{1}{1+\cdots}}=\frac{1+\sqrt{5}}{2}.$$

证 记

$$x_k=1+\cfrac{1}{1+\cfrac{1}{1+\cdots}},$$

则有递推式

$$x_{k+1} = 1 + \frac{1}{x_k}, \quad k = 0, 1, \cdots.$$

令 $\varphi(x) = 1 + \frac{1}{x}$, 则 $\varphi'(x) = -\frac{1}{x^2}$. 设 $\varphi(x)$ 有不动点 x^*, 即 $x^* = 1 + \frac{1}{x^*}$, 解之得

$$x^* = \frac{1+\sqrt{5}}{2}.$$

另一方面, 因

$$|\varphi'(x^*)| = \frac{1}{\left(\frac{1+\sqrt{5}}{2}\right)^2} < 1,$$

故由定理 7.3 知, $\{x_k\}$ 局部收敛于 x^*. □

为了刻画迭代序列 $\{x_k\}$ 的收敛速度, 我们引进收敛阶的概念, 它是衡量一个迭代算法优劣的重要指标之一.

定义 7.2 设 $\lim\limits_{k\to\infty} x_k = x^*$, 令 $e_k = x_k - x^*$, 如果存在某个实数 $p \geqslant 1$ 及常数 $c > 0$, 使得

$$\lim_{k\to\infty} \frac{|e_{k+1}|}{|e_k|^p} = c, \tag{7.11}$$

则称序列 $\{x_k\}$ 是 p 阶收敛的. 特别地, (1) 当 $p = 1$, $0 < c < 1$ 时, 称为线性收敛; (2) 当 $p = 1$, $c = 0$ 时, 称为超线性收敛; (3) 当 $p = 2$ 时, 称为平方收敛.

根据计算实践, 一般认为, 一个算法如果只有线性收敛速度, 那是认为不理想的, 有必要改进算法, 或采用加速技巧. 而一个算法如果具有超线性收敛速度, 那就认为是一个很不错的算法了. 至于构造具有平方敛速以上的算法, 则是数值分析人员梦寐以求的事情.

那么, 简单迭代法的收敛速度怎么样呢? 我们有下面的定理:

定理 7.4 设迭代函数 $\varphi(x)$ 满足:

(1) $x^* = \varphi(x^*)$, 且在 x^* 附近有 p 阶导数;

(2) $\varphi'(x^*) = \varphi''(x^*) = \cdots = \varphi^{(p-1)}(x^*) = 0$;

(3) $\varphi^{(p)}(x^*) \neq 0$,

那么, 迭代公式 (7.5) 是 p 阶收敛的.

证 由泰勒公式, 有

$$\begin{aligned} x_{k+1} &= \varphi(x_k) = \varphi(x^*) + \varphi'(x^*)(x_k - x^*) + \cdots \\ &\quad + \frac{\varphi^{p-1}(x^*)}{(p-1)!}(x_k - x^*)^{p-1} + \frac{\varphi^p(\xi_k)}{p!}(x_k - x^*)^p \\ &= x^* + \frac{\varphi^p(\xi_k)}{p!}(x_k - x^*)^p, \end{aligned}$$

其中 ξ_k 介于 x_k 与 x^* 之间, 所以

$$\frac{x_{k+1}-x^*}{(x_k-x^*)^p}=\frac{\varphi^p(\xi_k)}{p!}. \tag{7.12}$$

上式两边取极限, 并注意到当 $k\to\infty$ 时, $\xi_k\to x^*$, 即得

$$\lim_{k\to\infty}\frac{x_{k+1}-x^*}{(x_k-x^*)^p}=\frac{\varphi^p(x^*)}{p!},$$

即迭代公式 (7.5) 是 p 阶收敛的. □

例 7.6 *用简单迭代法求方程 $x^3-x^2-x-1=0$ 在区间 $[1,2]$ 上的一个根. 试用不同的方法构造迭代格式, 并指出每一格式是否收敛, 如果收敛指出收敛阶.*

解 *方法 1* 将原方程变为 $x=x^3-x^2-1$, 迭代公式为 $x_{k+1}=x_k^3-x_k^2-1$. 这里迭代函数 $\varphi(x)=x^3-x^2-1$, 有 $\varphi'(x)=3x^2-2x>1$, $\forall x\in(1,2]$. 若令 $x_0=2$, 有 $x_1=3$, $x_2=17,\cdots$, 迭代显然是不收敛的.

方法 2 将原方程变为 $x=\sqrt[3]{x^2+x+1}$, 迭代公式为 $x_{k+1}=\sqrt[3]{x_k^2+x_k+1}$. 这里迭代函数 $\varphi(x)=\sqrt[3]{x^2+x+1}$, 可以验证 $\varphi(x)\in[1,2]$, $\forall x\in[1,2]$, 且

$$\varphi'(x)=\frac{2x+1}{3\sqrt[3]{(x^2+x+1)^2}}<1,\quad \forall x\in[1,2].$$

故由定理 7.1 知这一迭代格式是收敛的. 由于 $\varphi'(x^*)\neq 0$, 故其收敛阶是 1 阶的.

方法 3 取迭代函数为

$$\varphi(x)=1+\frac{1}{x}+\frac{1}{x^2},$$

可以验证 $\varphi(x)\in[1,2]$, $\forall x\in[1,2]$, 且

$$\varphi'(x)=-\left(\frac{1}{x^2}+\frac{2}{x^3}\right).$$

故 $|\varphi'(x)|<1$, $\forall x\in[1,2]$. 由定理 7.1 知这一迭代格式也是收敛的. 显然 $\varphi'(x^*)\neq 0$, 故其收敛阶也是 1 阶的.

7.2.3 迭代法加速技巧

对于一个收敛的迭代过程, 只要迭代足够多次, 就可以使结果达到任意精度, 但有时迭代过程收敛缓慢, 从而使计算过程变得很大, 因此迭代过程的加速是个重要课题.

由定理 7.4 可知, 当 $\varphi'(x^*)\neq 0$ 时, 迭代公式 (7.5) 只有线性收敛速度, 收敛到真解的速度是比较缓慢的. 我们下面来考虑迭代过程的加速问题.

设 x_k 是根 x^* 的某个近似值, 用迭代公式校正一次得

$$y_k=\varphi(x_k).$$

假设 $\varphi'(x)$ 在所考察的范围内变化不大, 其估计值为 q, 则

$$x^* - y_k = \varphi(x^*) - \varphi(x_k) \approx q(x^* - x_k),$$

由此解出 x^* 得

$$x^* \approx \frac{1}{1-q}y_k - \frac{q}{1-q}x_k,$$

这就是说, 如果将迭代值 y_k 与 x_k 加权平均, 可望得到的

$$x_{k+1} = \frac{1}{1-q}y_k - \frac{q}{1-q}x_k$$

是比 y_k 更好的近似根. 这样加工后的计算过程是:

迭代 $y_k = \varphi(x_k)$,

改进 $x_{k+1} = \dfrac{1}{1-q}y_k - \dfrac{q}{1-q}x_k$.

这组迭代公式可以合并成

$$x_{k+1} = \frac{1}{1-q}[\varphi(x_k) - qx_k]. \tag{7.13}$$

例如, 用迭代公式 (7.13) 求解例 7.4, 由于 $\varphi'(x) = -\mathrm{e}^{-x}$, 故可取 $q = -0.6$, 则加速公式 (7.13) 的具体形式为

$$x_{k+1} = \frac{1}{1.6}(\mathrm{e}^{-x_k} + 0.6x_k).$$

利用简单迭代法通用程序 maiter.m, 在 MATLAB 命令窗口执行:

```
>> x=maiter(inline('(exp(-x)+0.6*x)/1.6'),0.5,1e-5)
```

可得到计算结果

```
k=3
x =
   0.56714328557022
```

即只需迭代 3 次, 就可以得到与例 7.4 同样精度的结果 (而在那里迭代了 17 次!), 可见这种加速的效果是明显的.

尽管如此, 加速迭代公式 (7.13) 使用起来却很不方便, 因为常数 q 一般是很难确定的. 鉴于此, Aitken-Steffensen (艾特肯–斯蒂文森) 提出了一个实用的加速迭代公式. 其基本思想是:

设当前近似点为 x_k, 令

$$y_k = \varphi(x_k), \tag{7.14}$$

$$z_k = \varphi(y_k). \tag{7.15}$$

仍设 $\varphi'(x) \approx q$, 于是有

$$y_k - x^* \approx q(x_k - x^*), \quad z_k - x^* \approx q(y_k - x^*),$$

将上面两式相除得

$$\frac{y_k - x^*}{z_k - x^*} \approx \frac{x_k - x^*}{y_k - x^*},$$

由上式解出 x^*, 得

$$x^* \approx \frac{x_k z_k - y_k^2}{z_k - 2y_k + x_k} = x_k - \frac{(y_k - x_k)^2}{z_k - 2y_k + x_k}.$$

可将上式的右端作为新的近似值, 即

$$x_{k+1} = x_k - \frac{(y_k - x_k)^2}{z_k - 2y_k + x_k}, \quad k = 0, 1, \cdots. \tag{7.16}$$

下面写出具体算法:

算法 7.4 (Aitken-Steffensen *加速方法*)

步 1 取初始点 x_0, 最大迭代次数 N 和精度要求 ε, 置 $k := 0$;

步 2 计算 $y_k = \varphi(x_k)$, $z_k = \varphi(y_k)$, 及

$$x_{k+1} = x_k - \frac{(y_k - x_k)^2}{z_k - 2y_k + x_k};$$

步 3 若 $|x_{k+1} - x_k| < \varepsilon$, 则停算;

步 4 若 $k = N$, 则停算; 否则, 置 $k := k + 1$, 转步 2.

可以证明, Aitken-Steffensen 加速方法具有平方收敛速度, 此结论略去不证.

根据算法 7.4, 编制 MATLAB 通用程序如下:

- Aitken-Steffensen *加速法* MATLAB *程序*

```
%maaitken.m
function x=maaitken(phi,x0,ep,N)
%用途: 用Aitken-Steffensen加速方法求f(x)=0的解
%格式: x=maaitken(phi,x0,ep,N)  phi为迭代函数, x0为迭代初值,
%ep为精度(默认1e-4), N为最大迭代次数(默认500), x返回近似根
if nargin<4,N=500;end
if nargin<3,ep=1e-4;end
k=0;
while k<N
    y=feval(phi,x0);
    z=feval(phi,y);
```

```
    x=x0-(y-x0)^2/(z-2*y+x0);
    if abs(x-x0)<ep, break;  end
    x0=x; k=k+1;
end
if k==N, warning('已达迭代次数上限'); end
disp(['k=',num2str(k)])
```

例 7.7 用 Aitken-Steffensen 加速法求方程 $f(x)=xe^x-1=0$ 在 $[0,1]$ 内的一个实根, 取初始点为 $x_0=0.5$, 精度为 10^{-5}.

解 在 MATLAB 命令窗口执行:

```
>> x=maaitken(inline('exp(-x)'),0.5,1e-5)
k= 2
x =
    0.56714329040978
```

即只需 2 次迭代就可以得到满足精度要求的结果.

例 7.8 设函数 $\varphi(x)$ 连续可微, 若迭代公式 $x_{k+1}=\varphi(x_k)$ 局部线性收敛, 对于 $a\in\mathbf{R}$, 构造迭代公式

$$x_{k+1}=\psi(x_k),\quad k=0,1,\cdots,\tag{7.17}$$

其中

$$\psi(x)=\frac{1}{1-a}\varphi(x)-\frac{a}{1-a}x.$$

试选取 a 的值使得迭代公式 (7.17) 具有更高的收敛阶.

解 因 $x_{k+1}=\varphi(x_k)$ 局部线性收敛, 故存在不动点 x^* 使得 $x^*=\varphi(x^*)$. 注意到

$$\psi(x^*)=\frac{1}{1-a}\varphi(x^*)-\frac{a}{1-a}x^*=\frac{1}{1-a}x^*-\frac{a}{1-a}x^*=x^*,$$

即 x^* 也是 $\psi(x)$ 的不动点. 对 $\psi(x)$ 求导数得

$$\psi'(x)=\frac{1}{1-a}\varphi'(x)-\frac{a}{1-a}.$$

要使迭代公式 (7.17) 具有比线性收敛更高的收敛阶, 必须满足条件 $\psi'(x^*)=0$, 即

$$\frac{1}{1-a}\varphi'(x^*)-\frac{a}{1-a}=0,$$

解得

$$a=\varphi(x^*).$$

因此, 如果选取 $a=\varphi(x^*)$, 则迭代公式 (7.17) 至少是二阶收敛的.

7.3 牛顿型方法

7.3.1 牛顿法的基本思想与算法

牛顿法是一种特殊形式的迭代法, 它是求解非线性方程最有效的方法之一. 其基本思想是: 利用泰勒公式将非线性函数在方程的某个近似根处展开, 然后截取其线性部分作为函数的一个近似, 通过解一个一元一次方程来获得原方程的一个新的近似根.

具体地说, 设当前点为 x_k, 将 $f(x)$ 在 x_k 处泰勒展开并截取线性部分得

$$f(x) \approx f(x_k) + f'(x_k)(x - x_k),$$

令上式右端为 0, 解得

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \quad k = 0, 1, \cdots. \tag{7.18}$$

式 (7.18) 称为牛顿迭代公式.

根据导数的几何意义及上述推导过程可知, 牛顿法的几何上表现为: x_{k+1} 是函数 $f(x)$ 在点 $(x_k, f(x_k))$ 处的切线与 x 轴的交点. 因此, 牛顿法的本质是一个不断用切线来近似曲线的过程, 故牛顿法也称为切线法.

至于牛顿法的终止条件, 可以采用与简单迭代法相同的终止条件 (因牛顿法本身就是迭代法), 于是我们可以写出算法过程如下.

算法 7.5 (牛顿法)

步 1 取初始点 x_0, 最大迭代次数 N 和精度要求 ε, 置 $k := 0$;

步 2 计算

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)};$$

步 3 若 $|x_{k+1} - x_k| < \varepsilon$, 则停算;

步 4 若 $k = N$, 则停算; 否则, 置 $k := k + 1$, 转步 2.

根据算法 7.5, 编制 MATLAB 通用程序如下:

- 牛顿法 MATLAB 程序

```
%manewton.m
function x=manewton(fun,dfun,x0,ep,N)
%用途:用牛顿法求解非线性方程f(x)=0
%格式:x=manewton(fun,dfun,x0,ep,N)  fun和dfun分别为表示f(x)和f'(x)
%的函数句柄, x0为迭代初值, ep为精度(默认1e-4), N为最大迭代
%次数(默认为500), x返回近似根
```

```
if nargin<5,N=500;end
if nargin<4,ep=1e-4;end
k=0;
while k<N
    x=x0-feval(fun,x0)/feval(dfun,x0);
    if abs(x-x0)<ep
        break;
    end
    x0=x; k=k+1;
end
if k==N, warning('已达迭代次数上限'); end
disp(['k=',num2str(k)])
```

例 7.9 用牛顿法程序 manewton.m 求方程 $f(x)=xe^x-1=0$ 在 $[0,1]$ 内的一个实根, 取初始点为 $x_0=0.5$, 精度为 10^{-5}.

解 在 MATLAB 命令窗口执行:

```
>> format long
>> fun=inline('x*exp(x)-1');
>> dfun=inline('(x+1)*exp(x)');
>> x=manewton(fun,dfun,0.5,1e-5)
```

得计算结果:

```
k=3
x =
   0.56714329040978
```

7.3.2 牛顿法的收敛速度

现在我们来分析牛顿法的收敛性及收敛速度. 我们有

定理 7.5 设函数 $f(x)$ 二次连续可导, x^* 满足 $f(x^*)=0$ 及 $f'(x^*)\neq 0$, 则存在 $\delta>0$, 当 $x_0\in[x_0-\delta,x_0+\delta]$ 时, 牛顿法是收敛的, 且收敛阶至少是 2 (即至少是平方收敛的).

证 不难发现, 牛顿法本质相当于迭代函数为

$$\varphi(x)=x-\frac{f(x)}{f'(x)}$$

的迭代法, 于是

$$\varphi'(x)=1-\frac{[f'(x)]^2-f(x)f''(x)}{[f'(x)]^2}=\frac{f(x)f''(x)}{[f'(x)]^2}.$$

由题设 $f(x^*)=0$ 及 $f'(x^*)\neq 0$ 可得 $\varphi'(x^*)=0$, 从而由定理 7.3 可知, 存在 $\delta>0$, 当 $x_0\in N(x^*,\delta)$ 时, 牛顿法收敛. 再由定理 7.4 可知, 其收敛阶至少是 2 阶的. □

由定理 7.5 的条件 ($f(x^*)=0,\ f'(x^*)\neq 0$) 可知, 当 x^* 是方程 $f(x)=0$ 的单根时, 收敛阶至少是 2 阶的. 如果 x^* 是方程 $f(x)=0$ 的重根的情形会是怎么样的呢? 我们来做一些简单的分析.

设 x^* 是方程 $f(x)=0$ 的 m 重根, 即

$$f(x)=(x-x^*)^m g(x),\quad m\geqslant 2,$$

其中 $g(x)$ 有二阶导数且 $g(x^*)\neq 0$, 计算 $\varphi(x)=x-f(x)/f'(x)$ 的导数, 得

$$\varphi'(x)=\frac{\left(1-\dfrac{1}{m}\right)+(x-x^*)\dfrac{2g'(x)}{mg(x)}+(x-x^*)^2\dfrac{g''(x)}{m^2g(x)}}{\left[1+(x-x^*)\dfrac{2g'(x)}{mg(x)}\right]^2},$$

所以有

$$\varphi'(x^*)=1-\frac{1}{m}.$$

这样, 当 $m\geqslant 2$ 时, $\varphi'(x^*)\neq 0$, 且有 $|\varphi'(x)|<1$, 这样, 牛顿法就至多只有线性收敛速度了. 这说明在重根的情形, 牛顿法失去了快速收敛的优点而变得不再实用.

为改善重根时牛顿法的收敛速度, 可以采用下面两种方法.

方法一 当根的重数 $m\geqslant 2$ 时, 将迭代函数改为

$$\varphi(x)=x-\frac{mf(x)}{f'(x)},\tag{7.19}$$

容易验证由上式定义的 $\varphi(x)$ 满足 $\varphi'(x^*)=0$, 因此迭代公式

$$x_{k+1}=x_k-\frac{mf(x_k)}{f'(x_k)},\quad k=0,1,\cdots\tag{7.20}$$

至少是二阶收敛的.

稍加思考我们便会发现上述加速方法并不适用, 因为事先并不知道根的重数 m, 故这一方法只具有理论上的意义, 下面的方法才是求重根时比较实用的加速方法.

方法二 若 x^* 是 $f(x)=0$ 的 m 重根, 则必为

$$\mu(x)=\frac{f(x)}{f'(x)}$$

的单根. 基于这个事实可以将牛顿迭代函数修改为

$$\varphi(x)=x-\frac{\mu(x)}{\mu'(x)}=x-\frac{f(x)f'(x)}{[f'(x)]^2-f(x)f''(x)}.$$

根据定理 7.5, 关于 $\mu(x)$ 的牛顿迭代公式

$$x_{k+1} = x_k - \frac{f(x_k)f'(x_k)}{[f'(x_k)]^2 - f(x_k)f''(x_k)}, \quad k = 0, 1, \cdots \tag{7.21}$$

至少是二阶收敛的. 公式 (7.21) 称为求重根的牛顿加速公式.

例 7.10 *取初始点为 $x_0 = 1.5$, 分别用牛顿法和牛顿加速公式 (7.21) 计算方程*

$$x^3 - x^2 - x + 1 = 0$$

的根.

解 容易发现 $x = 1$ 是二重根. 利用牛顿法的迭代公式为

$$x_{k+1} = x_k - \frac{x_k^2 - 1}{3x_k + 1}, \tag{7.22}$$

而利用牛顿加速公式 (7.21) 的迭代公式为

$$x_{k+1} = \frac{x_k^2 + 6x_k + 1}{3x_k^2 + 2x_k + 3}. \tag{7.23}$$

利用公式 (7.22), 迭代 13 次得近似解为：$x_{13} = 1.0001$. 而利用公式 (7.23) 迭代 3 次即可得到十分精确的结果：$x_1 = 0.96078$, $x_2 = 0.9996$, $x_3 = 1.0000$. 由此可见, 牛顿迭代加速公式 (7.21) 是有效的.

例 7.11 *构造计算 $\sqrt[3]{c}$ 的牛顿迭代公式, 并用此公式计算 $\sqrt[3]{412}$ 的近似值, 精确到 10^{-8}.*

解 令 $x = \sqrt[3]{c}$, 即求等价方程 $x^3 - c = 0$ 的根. 取 $f(x) = x^3 - c$, 则 $f'(x) = 3x^2$. 代入牛顿迭代公式有

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} = x_k - \frac{x_k^3 - c}{3x_k^2} = \frac{1}{3}\left(2x_k + \frac{c}{x_k^2}\right).$$

计算 $\sqrt[3]{412}$ 时, 因 $f(7) < 0$, $f(8) > 0$, 故 $x^* \in [7, 8]$. 取 $x_0 = 7$, 利用上述迭代公式进行计算, 计算结果下表.

k	x_k	k	x_k
0	7	3	7.441018862
1	7.469387755	4	7.441018861
2	7.441126470		

可以看出, $\sqrt[3]{412}$ 精确到 10^{-8} 的值为 7.44101886.

7.3.3 阻尼牛顿法

一般来说，牛顿法的收敛性依赖于初值 x_0 的选取，如果 x_0 偏离 x^* 较远，则牛顿法可能收敛缓慢甚至发散. 例如，用牛顿法求方程 $x^3-x-1=0$ 的近似根，如果取 $x_0=1.5$，用牛顿迭代公式

$$x_{k+1}=x_k-\frac{x_k^3-x_k-1}{3x_k^2-1} \tag{7.24}$$

迭代 3 次可得结果：$x_1=1.3478$, $x_2=1.3252$, $x_3=1.3247$，其误差小于 10^{-5}. 但如果取 $x_0=-2.0$，则要得到同样精度的解需要迭代 65 次!

因此，为了保证当 x_0 远离 x^* 时，迭代仍然收敛，可在牛顿迭代公式中增加一个参数 α，改为

$$x_{k+1}=x_k-\alpha_k\frac{f(x_k)}{f'(x_k)},\quad k=0,1,\cdots, \tag{7.25}$$

其中 α_k 的选择保证

$$|f(x_{k+1})|<|f(x_k)|. \tag{7.26}$$

公式 (7.25) 和 (7.26) 合起来称为阻尼牛顿法或牛顿下降法.

如何选择 α_k，通常采用简单后退准则，即取 $\rho=0.5$，记 m_k 是使下面不等式成立的最小非负整数 m：

$$\left|f\big(x_k-\rho^m f(x_k)/f'(x_k)\big)\right|<|f(x_k)|, \tag{7.27}$$

然后令 $\alpha_k=\rho^{m_k}$ 即可.

现写出阻尼牛顿法具体的算法步骤如下：

算法 7.6 (阻尼牛顿法)

步 1 取初始点 x_0, $\rho=0.5$，最大迭代次数 N 和精度要求 ε，置 $k:=0$;

步 2 计算 $f(x_k)$ 及 $f'(x_k)$;

步 3 对于 $m=0,1,\cdots$，检验下面的不等式：

$$\left|f\big(x_k-\rho^m f(x_k)/f'(x_k)\big)\right|<|f(x_k)|,$$

记 m_k 为使上述不等式成立的最小非负整数 m;

步 4 置

$$\alpha_k=\rho^{m_k},\quad x_{k+1}=x_k-\alpha_k f(x_k)/f'(x_k);$$

步 5 若 $|x_{k+1}-x_k|<\varepsilon$，则停算;

步 6 若 $k=N$，则停算；否则，置 $k:=k+1$，转步 2.

7.3.4 离散牛顿法

用牛顿法或阻尼牛顿法解方程 $f(x)=0$ 的优点是收敛速度快, 但牛顿法有一个明显的缺点: 每次迭代除需计算函数值 $f(x_k)$, 还需计算导数 $f'(x_k)$ 的值, 如果 $f(x)$ 比较复杂, 计算 $f'(x_k)$ 就可能十分麻烦. 尤其当 $|f'(x_k)|$ 很小时, 计算需十分精确, 否则会产生较大的误差.

为避开计算导数, 可以改用差商 (离散形式) 代替导数 (连续形式), 即

$$f'(x_k) \approx \frac{f(x_k)-f(x_{k-1})}{x_k - x_{k-1}},$$

得到牛顿迭代公式 (7.5) 的离散化形式

$$x_{k+1} = x_k - \frac{f(x_k)}{f(x_k)-f(x_{k-1})}(x_k - x_{k-1}). \tag{7.28}$$

迭代公式 (7.28) 称为离散牛顿法或割线法. 可以证明下面的收敛定理.

定理 7.6 设函数 $f(x)$ 在其零点 x^* 的某个邻域 $S=\{x||x-x^*|\leqslant \delta\}$ 内有二阶连续导数, 且对任意 $x\in S$, 有 $f'(x)\neq 0$, 则当 $\delta>0$ 充分小时, 对 S 中任意 x_0, x_1, 由离散牛顿迭代 (7.28) 产生的序列 $\{x_k\}$ 收敛到方程 $f(x)=0$ 的根 x^*, 且具有超线性收敛速度, 其收敛阶 $p\approx 1.618$.

证明可参见文献 [3] 第 348~350 页.

由于离散牛顿法不需要计算导数, 虽然收敛阶低于牛顿法, 但高于简单迭代法. 因此, 离散牛顿法在非线性方程的求解中得到广泛的应用, 也是工程计算中的常用方法之一.

综上所述, 离散牛顿法的计算步骤可归纳如下.

算法 7.7 (离散牛顿法)

步 1 取初始点 x_0, x_1, 最大迭代次数 N 和精度要求 ε, 置 $k:=0$;

步 2 计算 $f(x_k)$ 及 $f(x_{k-1})$;

步 3 置

$$x_{k+1} := x_k - \frac{f(x_k)}{f(x_k)-f(x_{k-1})}(x_k - x_{k-1});$$

步 4 若 $|x_{k+1}-x_k|<\varepsilon$, 则停算;

步 5 若 $k=N$, 则停算; 否则, 置 $x_{k-1}:=x_k$, $x_k:=x_{k+1}$, $k:=k+1$, 转步 2.

根据算法 7.7, 编制 MATLAB 通用程序如下:

- 离散牛顿法 MATLAB 程序

```
%maqnewt.m
function x=maqnewt(fun,x0,x1,ep,N)
%用途:用离散牛顿法求解非线性方程f(x)=0
```

```
%格式:x=maqnewt(fun,x0,x1,ep,N)  fun为表示f(x)的函数句
%柄,x0, x1为迭代初值, ep为精度(默认1e-4), N为最大迭代次
%数(默认为500), x返回近似根
if nargin<5,N=500;end
if nargin<4,ep=1e-4;end
k=0;
while k<N
  x=x1-(x1-x0)*feval(fun,x1)/(feval(fun,x1)-feval(fun,x0));
  if abs(x-x1)<ep
    break;
  end
  x0=x1;x1=x;
  k=k+1;
end
if k==N, warning('已达迭代次数上限'); end
disp(['k=',num2str(k)])
```

例 7.12 用离散牛顿法程序 maqnewt.m, 求方程 $f(x) = xe^x - 1 = 0$ 在 $[0,1]$ 内的一个实根, 取初始点为 $x_0 = 0.4$, $x_1 = 0.6$, 精度为 10^{-5}.

解 在 MATLAB 命令窗口执行:

```
>> x=maqnewt(inline('x*exp(x)-1'),0.4,0.6,1e-5)
```

得计算结果:

```
k=3
x =
   0.56714329035989
```

习 题 7

(I) 理论分析题

7.1 已知方程 $x^3 + x - 4 = 0$ 在区间 $[1,2]$ 内有一根, 试问用二分法求根, 使其具有 5 位有效数字至少应二分多少次?

7.2 基于迭代原理证明

$$\sqrt{1+\sqrt{1+\sqrt{1+\cdots}}} = \frac{1+\sqrt{5}}{2}.$$

7.3 利用适当的迭代格式证明

$$\lim_{k\to\infty}\underbrace{\sqrt{2+\sqrt{2+\cdots+\sqrt{2}}}}_{k\text{个}2}=2.$$

7.4 给定函数 $f(x)$, 设对一切 x, $f'(x)$ 存在且 $0<m\leqslant f'(x)\leqslant M$, 试证明: $\forall\beta\in(0,2/M)$, 迭代过程

$$x_{k+1}=x_k-\beta f(x_k)$$

均收敛于方程 $f(x)=0$ 的根 x^*.

7.5 设 $\varphi(x)=x^2-2x+2$, 问取什么初值 x_0, 由 $x_{k+1}=\varphi(x_k)$ 产生的序列 $\{x_k\}$ 收敛到 φ 的不动点? 并给出图形解释.

7.6 对方程 (1) $x-\cos x=0$, (2) $3x^2-\mathrm{e}^x=0$, 确定 $[a,b]$ 及 φ, 使 $x_{k+1}=\varphi(x_k)$ 对任意 $x_0\in[a,b]$ 均收敛, 并求出方程的各个根, 误差不超过 10^{-3}.

7.7 已知 $x=\varphi(x)$ 在 $[a,\ b]$ 上只有一个实根, 且当 $x\in[a,\ b]$ 时, $|\varphi'(x)|\geqslant L>1$ (L 为常数), 问如何将 $x=\varphi(x)$ 化为适合于迭代的形式.

7.8 证明对任何初始值 $x_0\in\mathbf{R}$, 由迭代公式

$$x_{k+1}=\cos x_k,\quad k=0,1,2\cdots,$$

则所产生的序列 $\{x_k\}_{k=0}^{\infty}$ 都收敛于方程 $x=\cos x$ 的根.

7.9 设方程 $2x-\sin x-4=0$ 的迭代格式 $x_{k+1}=2+0.5\sin x_k$.

(1) 证明对任意的 $x_0\in\mathbf{R}$, 均有 $\{x_k\}\to x^*$ $(k\to\infty)$, 其中 x^* 是方程的根;

(2) 取 $x_0=2$, 求此方程的近似根, 使误差不拆过 10^{-3};

(3) 证明此迭代法是线性收敛的.

7.10 取初始值 $x_0=3$, 用简单迭代法求解方程 $\ln x-x+2=0$ 在 $(2,\infty)$ 内的根, 并用斯蒂文森迭代法加速.

7.11 设 x^* 是方程 $f(x)=0$ 的单根, $x=\varphi(x)$ 是 $f(x)=0$ 的等价方程. 若 $\varphi(x)=x-m(x)f(x)$, 证明当 $m(x^*)\neq\dfrac{1}{f'(x^*)}$ 时, $x_{k+1}=\varphi(x_k)$ 至多是一阶收敛的; 当 $m(x^*)=\dfrac{1}{f'(x^*)}$ 时, $x_{k+1}=\varphi(x_k)$ 至少是二阶收敛的.

7.12 为了简化计算, 可在牛顿迭代公式中用 $f'(x_0)$ 取代 $f'(x_k)$, 试问这种迭代格式的收敛阶是几阶的?

7.13 对于实数 $a\neq0$, 给出一个不用除法运算求其倒数的二阶收敛迭代公式, 并分析初值 x_0 可以选取的范围.

7.14 导出计算 $\dfrac{1}{\sqrt{a}}$ (其中 $a>0$) 的牛顿迭代公式, 要求该迭代公式既无开方又无除法运算.

7.15 将牛顿法用于求解 $x^3-10=0$, 讨论取什么初值可使迭代收敛.

7.16 证明: 解方程 $(x^2-a)^2=0$ 求 $\sqrt{a}$ 的牛顿法

$$x_{k+1}=\frac{3}{4}x_k+\frac{a}{4x_k}$$

仅为线性收敛.

7.17 设牛顿法的迭代序列 $\{x_k\}$ 收敛到方程 $f(x)=0$ 的某个单根 x^*, 证明：误差界 $e_k = x_k - x^*$ 至少是 2 阶的, 即

$$\lim_{k\to\infty}\frac{e_{k+1}}{e_k}=\frac{f''(x^*)}{2f'(x^*)}.$$

7.18 设 x^* 是方程 $f(x)=0$ 的 $m(\geqslant 2)$ 重根, 证明修正的牛顿迭代

$$x_{k+1}=x_k-m\frac{f(x_k)}{f'(x_k)}$$

为平方收敛.

7.19 证明：Steffenson 方法

$$x_{k+1}=x_k-\frac{[f(x_k)]^2}{f(x_k+f(x_k))-f(x_k)}$$

对于 $f(x)=0$ 的单根 x^* 为平方收敛.

7.20 证明：迭代公式

$$x_{k+1}=\frac{2x_k(x_k^2+a)}{3x_k^2+a}$$

是计算 $\sqrt{a}$ 的 3 阶方法. 假定初值 x_0 充分靠近根 x^*, 求

$$\lim_{k\to\infty}\frac{x_{k+1}-\sqrt{a}}{(x_k-\sqrt{a})^3}$$

的值.

7.21 试确定常数 α, β, γ, 使迭代公式

$$x_{k+1}=\alpha x_k+\frac{\beta a}{x_k^2}+\frac{\gamma a^2}{x_k^5}$$

产生的序列收敛到 $\sqrt[3]{a}$, 并使其收敛阶尽量高.

7.22 设 x^* 是 $f(x)=0$ 的单根, f 二次连续可导, 试证明下面的迭代公式

$$\begin{cases} y_k = x_k-\dfrac{f(x_k)}{f'(x_k)}, \\ x_{k+1}=y_k-\dfrac{f(y_k)}{f'(x_k)}, \end{cases} \quad k=0,1,\cdots$$

至少三阶收敛.

7.23 设 $\varphi = x-p(x)f(x)-q(x)f^2(x)$, 试确定函数 $p(x)$, $q(x)$, 使求解 $f(x)=0$ 且以 $\varphi(x)$ 为迭代函数的迭代法至少三阶收敛.

(Ⅱ) 上机实验题

7.1 利用算法 7.1, 编制 MATLAB 程序, 求方程

$$x^3-3.2x^2+1.9x+0.8=0$$

的隔根区间.

7.2 利用算法 7.2 (二分法), 编制 MATLAB 程序, 求方程

$$\cos x - 3x + 1 = 0$$

在 $[0,1]$ 内的一个根, 精度为 10^{-5}.

7.3 适当选取迭代格式, 利用算法 7.3 (简单迭代法), 编制 MATLAB 程序, 求方程

$$10^x - x - 2 = 0$$

在 $[0.3, 0.4]$ 内的一个根, 精度为 10^{-5}.

7.4 编制 MATLAB 程序, 比较算法 7.3 和算法 7.4 (迭代加速法) 在求解方程

$$x^3 - x - 1 = 0$$

的计算效率. 隔根区间为 $[1,2]$, 精度为 10^{-5}.

7.5 利用算法 7.5 (牛顿法), 编制 MATLAB 程序, 求方程

$$x^5 - x - 0.2 = 0$$

在 $[0,1]$ 内的一个根, 精度为 10^{-5}.

7.6 利用算法 7.7 (离散牛顿法), 编制 MATLAB 程序, 求方程

$$e^x + 10x - 2 = 0$$

在 $[0,1]$ 内的一个根, 精度为 10^{-5}.

第 8 章　矩阵特征值问题的计算 ①

许多工程实际问题的求解，如振动问题、稳定性问题等，最终都归结为求某些矩阵的特征值和特征向量的问题. 大家知道，n 阶方阵 A 的特征值与特征向量，是满足如下两个方程的数 λ 和非零向量 ξ：

$$|\lambda I - A| = 0, \tag{8.1}$$

$$A\xi = \lambda\xi \quad \text{或} \quad (\lambda I - A)\xi = 0. \tag{8.2}$$

方程 (8.1) 称为矩阵 A 的特征方程，它是 λ 的 n 次代数方程，当 n 较大时难以准确求解. 因此，从数值计算的观点来看，用特征多项式来求矩阵特征值的方法并不可取.

在实际应用中，求矩阵的特征值和特征向量通常采用迭代法. 其基本思想是，将特征值和特征向量作为一个无限序列的极限来求得. 舍入误差对这类方法的影响很小，但通常计算量较大.

根据具体问题的需要，有些实际问题只要计算绝对值最大的特征值，也有些实际问题则只需计算绝对值最小的特征值，当然，更多的问题则要求计算全部特征值和特征向量. 本章介绍几种目前在计算机上比较常用的矩阵特征值问题的数值方法.

8.1 幂法和反幂法

8.1.1 幂法及其通用程序

幂法是通过求矩阵的特征向量来求出特征值的一种迭代法. 它主要用来求按模最大的特征值和相应的特征向量的. 其优点是算法简单，容易计算机实现，缺点是收敛速度慢，其有效性依赖于矩阵特征值的分布情况.

适于使用幂法的常见情形是：A 的特征值可按模的大小排列为 $|\lambda_1| > |\lambda_2| \geqslant |\lambda_3| \geqslant \cdots \geqslant |\lambda_n|$，且其对应特征向量 $\xi_1, \xi_2, \cdots, \xi_n$ 线性无关. 此时，任意非零向量 x_0 均可用 $\xi_1, \xi_2, \cdots, \xi_n$ 线性表示，即

$$x_0 = \beta_1\xi_1 + \beta_2\xi_2 + \cdots + \beta_n\xi_n, \tag{8.3}$$

① 教学计划课时在 60 学时以下者可不讲授本章.

且 $\beta_1, \beta_2, \cdots, \beta_n$ 不全为零. 作向量序列 $x_k = A^k x_0$, 则

$$\begin{aligned} x_k = A^k x_0 &= \beta_1 A^k \xi_1 + \beta_2 A^k \xi_2 + \cdots + \beta_n A^k \xi_n \\ &= \beta_1 \lambda_1^k \xi_1 + \beta_2 \lambda_2^k \xi_2 + \cdots + \beta_n \lambda_n^k \xi_n \\ &= \lambda_1 \left[\beta_1 \xi_1 + \beta_2 \left(\frac{\lambda_2}{\lambda_1} \right)^k \xi_2 + \cdots + \beta_n \left(\frac{\lambda_n}{\lambda_1} \right)^k \xi_n \right]. \end{aligned}$$

由此可见, 若 $\beta_1 \neq 0$, 则因 $k \to \infty$ 时,

$$\left(\frac{\lambda_i}{\lambda_1} \right)^k \to 0 \quad (i = 2, \cdots, n),$$

故当 k 充分大时, 必有

$$x_k \approx \lambda_1^k \beta_1 \xi_1,$$

即 x_k 可以近似看成 λ_1 对应的特征向量; 而 x_k 与 x_{k-1} 分量之比

$$\frac{(x_k)_i}{(x_{k-1})_i} \approx \frac{\lambda_1^k \beta_1 (\xi_1)_i}{\lambda_1^{k-1} \beta_1 (\xi_1)_i} = \lambda_1.$$

于是利用向量序列 $\{x_k\}$ 既可求出按模最大的特征值 λ_1, 又可求出对应的特征向量 ξ_1.

在实际计算中, 考虑到当 $|\lambda_1| > 1$ 时, $\lambda_1^k \to \infty$; 当 $|\lambda_1| < 1$ 时, $\lambda_1^k \to 0$, 因而计算 x_k 时可能会导致计算机“上溢”或“下溢”现象发生, 故采取每步将 x_k 归一化处理的办法, 即将 x_k 的各分量都除以绝对值最大的分量, 使 $\|x_k\|_\infty = 1$. 于是, 求 A 按模最大特征值 λ_1 和对应的特征向量 ξ_1 的算法, 可归纳为如下步骤:

算法 8.1 (幂法)

步 1 输入矩阵 A, 初始向量 y_0, 误差限 ε, 最大迭代次数 N;

步 2 置 $k := 1$, $\mu := 0$, $x_0 = y_0 / \|y_0\|_\infty$;

步 3 计算 $y_k = A x_{k-1}$;

步 4 计算

$$|[y_k]_r| = \max_{1 \leqslant i \leqslant n} |[y_k]_i|,$$

其中 $[y_k]_i$ 表示向量 y_k 的第 i 个分量, 并置

$$m_k := [y_k]_r, \quad x_k := y_k / m_k;$$

步 5 若 $|m_k - \mu| < \varepsilon$, 停算, 输出 m_k, x_k; 否则, 转步 6;

步 6 若 $k < N$, 置 $k := k + 1$, $\mu := m_k$, 转步 3; 否则输出计算失败信息, 停算.

上述算法称为幂法. 可以证明下面的收敛性定理:

定理 8.1 设矩阵 A 的特征值可按模的大小排列为 $|\lambda_1|>|\lambda_2|\geqslant|\lambda_3|\geqslant\cdots\geqslant|\lambda_n|$, 且其对应特征向量 $\xi_1,\xi_2,\cdots,\xi_n$ 线性无关. 序列 $\{x_k\}$ 由算法 8.1 产生, 则有

$$\lim_{k\to\infty}x_k=\xi_1^0=\frac{\xi_1}{\max(\xi_1)},\quad \lim_{k\to\infty}m_k=\lambda_1, \tag{8.4}$$

其中 ξ_1^0 表示将 ξ_1 单位化后得到的向量, $\max(\xi_1)$ 表示向量 ξ_1 绝对值最大的分量.

证 由算法 8.1 步 3 和步 4 知

$$x_k=\frac{y_k}{m_k}=\frac{Ax_{k-1}}{m_k}=\frac{A^2x_{k-2}}{m_km_{k-1}}=\cdots=\frac{A^kx_0}{m_km_{k-1}\cdots m_1}.$$

由于 x_k 的最大分量为 1, 即 $\max(x_k)=1$, 故

$$m_km_{k-1}\cdots m_1=\max(A^kx_0).$$

从而

$$x_k=\frac{A^kx_0}{\max(A^kx_0)}=\frac{\lambda_1^k\left[\beta_1\xi_1+\sum\limits_{i=2}^n\beta_i\left(\frac{\lambda_i}{\lambda_1}\right)^k\xi_i\right]}{\max\left\{\lambda_1^k\left[\beta_1\xi_1+\sum\limits_{i=2}^n\beta_i\left(\frac{\lambda_i}{\lambda_1}\right)^k\xi_i\right]\right\}}$$

$$=\frac{\beta_1\xi_1+\sum\limits_{i=2}^n\beta_i\left(\frac{\lambda_i}{\lambda_1}\right)^k\xi_i}{\max\left\{\beta_1\xi_1+\sum\limits_{i=2}^n\beta_i\left(\frac{\lambda_i}{\lambda_1}\right)^k\xi_i\right\}}.$$

可见

$$\lim_{k\to\infty}x_k=\frac{\beta_1\xi_1}{\max(\beta_1\xi_1)}=\frac{\xi_1}{\max(\xi_1)}=\xi_1^0.$$

又

$$y_k=Ax_{k-1}=\frac{A^kx_0}{\max(A^{k-1}x_0)}$$

$$=\frac{\lambda_1^k\left[\beta_1\xi_1+\sum\limits_{i=2}^n\beta_i\left(\frac{\lambda_i}{\lambda_1}\right)^k\xi_i\right]}{\lambda_1^{k-1}\max\left[\beta_1\xi_1+\sum\limits_{i=2}^n\beta_i\left(\frac{\lambda_i}{\lambda_1}\right)^{k-1}\xi_i\right]},$$

即有

$$m_k=\max(y_k)=\lambda_1\frac{\max\left[\beta_1\xi_1+\sum\limits_{i=2}^n\beta_i\left(\frac{\lambda_i}{\lambda_1}\right)^k\xi_i\right]}{\max\left[\beta_1\xi_1+\sum\limits_{i=2}^n\beta_i\left(\frac{\lambda_i}{\lambda_1}\right)^{k-1}\xi_i\right]},$$

从而

$$\lim_{k\to\infty} m_k = \lambda_1$$

成立. □

下面给出幂法的 MATLAB 通用程序.

- 幂法 MATLAB 程序

```
%mapower.m
function [lambda,v,k]=mapower(A,x0,epsilon,max1)
%用途: 用幂法求矩阵的模最大特征值和对应的特征向量
%格式: [lambda,v,k]=mapower(A,x0,epsilon,max1)  A为n阶方阵, x0为
%初始向量, epsilon为上限, max1为循环次数. lambda返回按模最大
%的特征值, v返回对应的特征向量, k返回迭代次数.
lambda=0; k=0; err=1;
while((k<max1)&(err>epsilon))
   y=A*x;  [m,j]=max(abs(y));
   m=y(j); x=y/m;
   err=abs(lambda-m);
   lambda=m;  k=k+1;
end
v=x;
```

例 8.1 利用幂法通用程序 mapower.m, 求 A 按模最大的特征值 λ_1 和对应的特征向量 ξ_1, 其中

$$A = \begin{pmatrix} -1 & 2 & 1 \\ 2 & -4 & 1 \\ 1 & 1 & -6 \end{pmatrix}.$$

解 在 MATLAB 命令窗口执行:

```
>> A=[-1,2,1;2,-4,1;1,1,-6];
>> x=[1;1;1];
>> [lambda,v,k]=mapower(A,x,1e-10,100)
lambda =
    -6.42106661402299
v =
    -0.04614548326375
    -0.37492113082845
     1.00000000000000
```

```
k =
      78
```

8.1.2 幂法的加速技术

可以证明在定理 8.1 的条件下, 算法 8.1 是线性收敛的. 事实上, 设 k 充分大时, A^kx_0 的最大分量是它的第 j 个分量, 则

$$
\begin{aligned}
m_k-\lambda_1 &= \max(y_k)-\lambda_1=\frac{A^kx_0}{\max(A^{k-1}x_0)}-\lambda_1\\
&=\frac{\left[\beta_1\lambda_1^k\xi_1+\beta_2\lambda_2^k\xi_2+\cdots+\beta_2\lambda_n^k\xi_n\right]_j}{\left[\beta_1\lambda_1^{k-1}\xi_1+\beta_2\lambda_2^{k-1}\xi_2+\cdots+\beta_n\lambda_n^{k-1}\xi_n\right]_j}-\lambda_1\\
&=\frac{\left[\beta_2\lambda_2^{k-1}(\lambda_2-\lambda_1)\xi_2+\cdots+\beta_n\lambda_n^{k-1}(\lambda_n-\lambda_1)\xi_n\right]_j}{\left[\beta_1\lambda_1^{k-1}\xi_1+\beta_2\lambda_2^{k-1}\xi_2+\cdots+\beta_n\lambda_n^{k-1}\xi_n\right]_j},
\end{aligned}
$$

于是有

$$
\begin{aligned}
m_k-\lambda_1 &= \left(\frac{\lambda_2}{\lambda_1}\right)^{k-1}\frac{\left[\beta_2(\lambda_2-\lambda_1)\xi_2+\sum\limits_{i=3}^{n}\beta_i\left(\frac{\lambda_i}{\lambda_2}\right)^{k-1}(\lambda_n-\lambda_1)\xi_i\right]_j}{\left[\beta_1\xi_1+\sum\limits_{i=2}^{n}\beta_i\left(\frac{\lambda_i}{\lambda_1}\right)^{k-1}\xi_i\right]_j}\\
&=\left(\frac{\lambda_2}{\lambda_1}\right)^{k-1}M_k,\quad M_k\to M,
\end{aligned}
$$

其中 M 为常数. 所以, 当 $k\to\infty$ 时,

$$
\frac{|m_{k+1}-\lambda_1|}{|m_k-\lambda_1|}=\left|\frac{M_{k+1}(\lambda_2/\lambda_1)^k}{M_k(\lambda_2/\lambda_1)^{k-1}}\right|\to\left|\frac{\lambda_2}{\lambda_1}\right|,
$$

即幂法的收敛速度与比值 $|\lambda_2/\lambda_1|$ 的大小有关, $|\lambda_2/\lambda_1|$ 越小, 收敛速度越快, 当此比值接近于 1 时, 收敛速度是非常缓慢的. 这一事实启发我们, 可以对矩阵作一圆点位移, 令

$$
B=A-\alpha I,
$$

其中, α 为参数, 选择此参数可使矩阵 B 的上述比值更小, 以加快幂法的收敛速度. 设矩阵 A 的特征值为 $\lambda_1,\lambda_2,\cdots,\lambda_n$, 对应的特征向量为 $\xi_1,\xi_2,\cdots,\xi_n$, 则矩阵 B 的特征值为 $\lambda_1-\alpha,\lambda_2-\alpha,\cdots,\lambda_n-\alpha$, B 的特征向量和 A 的特征向量相同. 假设原点位移后, B 的特征值 $\lambda_1-\alpha$ 仍为绝对值最大的特征值, 选择 α 的目的是使

$$
\max_{2\leqslant i\leqslant n}\frac{|\lambda_i-\alpha|}{|\lambda_1-\alpha|}<\left|\frac{\lambda_2}{\lambda_1}\right|. \tag{8.5}
$$

适当地选择 α 可使幂法的收敛速度得到加速. 此时 $m_k \to \lambda_1 - \alpha$, $m_k + \alpha \to \lambda_1$, 而 x_k 仍然收敛于 A 的特征向量 ξ_1^0. 这种加速收敛的方法称为原点位移法.

在实际计算中, 由于事先矩阵的特征值分布情况一般是不知道的, 参数 α 的选取存在困难, 故原点位移法是很难实现的. 但是我们将会看到, 在反幂法中原点位移参数 α 是非常容易选取的, 因此, 带原点位移的反幂法已成为改进特征值和特征向量精度的标准算法. 我们给出采用原点加速技术的幂法 MATLAB 通用程序如下:

- *原点位移幂法 MATLAB 程序*

```
%mapowerp.m
function [lambda,v,k]=mapowerp(A,x,alpha,epsilon,max1)
%用途: 用原点位移幂法求矩阵的模最大特征值和对应的特征向量
%格式: [lambda,v,k]=mapowerp(A,x,alpha, epsilon,max1)
%A为n阶方阵, x为初始向量, epsilon为上限, max1为循环次数,
%alpha为原点位移参数. lambda返回按模最大的特征值, v返回对
%应的特征向量,k返回迭代次数.
lambda=0; k=0; err=1;
n=length(x); A=A-alpha*eye(n);
while((k<max1)&(err>epsilon))
    y=A*x; [m,j]=max(abs(y));
    m=y(j); x=y/m;
    err=abs(lambda-m);
    lambda=m; k=k+1;
end
lambda=lambda-alpha; v=x;
```

例 8.2 *利用原点位移幂法通用程序 mapowerp.m, 求 A 按模最大的特征值 λ_1 和对应的特征向量 ξ_1, 其中*

$$A = \begin{pmatrix} -1 & 2 & 1 \\ 2 & -4 & 1 \\ 1 & 1 & -6 \end{pmatrix}.$$

解 选取 $\alpha = -2$, 在 MATLAB 命令窗口执行:

```
>> A=[-1,2,1;2,-4,1;1,1,-6];
>> x=[1;1;1];
>> [lambda,v,k]=mapowerp(A,x,-2,1e-10,100)
lambda =
```

```
        -6.42106661417413
v =
     -0.04614548312203
     -0.37492113109948
      1.00000000000000
k =
     52
```

由上述结果可以看出, 在同样的精度控制下, 带原点位移加速的幂法只需要迭代 52 次, 而纯粹的幂法则需要迭代 78 次 (例 8.1).

8.1.3 反幂法及其通用程序

设 A 可逆, 则对 A 的逆阵 A^{-1} 施以幂法称为反幂法. 由于 $A\xi_i = \lambda_i\xi_i$ 时, 成立 $A^{-1}\xi_i = \lambda_i^{-1}\xi_i$. 因此, 若 $|\lambda_1| \geqslant |\lambda_2| \geqslant \cdots \geqslant |\lambda_{n-1}| > |\lambda_n|$, 则 λ_n^{-1} 是 A^{-1} 按模最大的特征值, 此时按反幂法, 必有

$$m_k \to \lambda_n^{-1}, \quad x_k \to \xi_n^0,$$

且其收敛率为 $|\lambda_n/\lambda_{n-1}|$. 任取初始向量 x_0, 构造向量序列

$$x_{k+1} = A^{-1}x_k, \quad k = 0, 1, 2, \cdots, \tag{8.6}$$

按幂法计算即可. 但用 (8.6) 计算, 首先要求 A^{-1}, 这比较麻烦而且是不经济的. 实际计算中, 通常用解方程组的办法, 即用

$$Ax_{k+1} = x_k, \quad k = 0, 1, 2, \cdots \tag{8.7}$$

求 x_{k+1}. 为防止计算机溢出, 实际计算时所用的公式为

$$\begin{cases} y_k = x_k/\|x_k\|_\infty, \\ Ax_{k+1} = y_k, \end{cases} \quad k = 0, 1, 2, \cdots. \tag{8.8}$$

反幂法主要用于已知矩阵的近似特征值为 α 时, 求矩阵的特征向量并提高特征值的精度. 此时, 可以用原点位移法来加速迭代过程, 于是 (8.8) 相应为

$$\begin{cases} y_k = x_k/\|x_k\|_\infty, \\ (A - \alpha I)x_{k+1} = y_k, \end{cases} \quad k = 0, 1, 2, \cdots. \tag{8.9}$$

为节省计算量, 通常先用列主元 LU 分解将矩阵 $A - \alpha I$ 分解为下三角矩阵 L 和上三角矩阵 U, 这样在迭代过程中每一步就只要解两个三角方程组了.

反幂法的计算步骤如下:

算法 8.2（反幂法）

步 1 输入矩阵 A, 初始向量 x_0, 近似值 α, 误差限 ε, 最大迭代次数 N;

步 2 置 $k:=1,\ \mu:=1$;

步 3 作列主元 LU 分解 $P(A-\alpha I)=LU$;

步 4 计算 $|[x_k]_r|=\max\limits_{1\leqslant i\leqslant n}|[x_k]_i|$, 并置 $m:=[x_k]_r$;

步 5 计算 x 的新值: $y=x/m,\ \ Lz=Py,\ \ Ux=z$;

步 6 若 $\left|\dfrac{1}{m}-\dfrac{1}{\mu}\right|<\varepsilon$, 则置 $\lambda:=\alpha+\dfrac{1}{m}$, 输出 λ, x, 停算; 否则, 转步 7.

步 7 若 $k<N$, 置 $k:=k+1$, $\mu:=m$, 转步 4; 否则输出计算失败信息, 停算.

注 8.1 (1) 在上述算法中, 如果 $\alpha=0$, 则求出 A 的按模最小的特征值.

(2) 通常首先用幂法求出 A 的按模最大的近似特征值作为算法 8.2 中的 α 值, 再使用该算法对 α 和相应的特征向量进行精确化.

我们给出反幂法的 MATLAB 通用程序如下:

```
%mainvp.m
function [lambda,v,k]=mainvp(A,x,alpha,epsilon,max1)
%用途: 用原点位移加速反幂法求矩阵的模最大特征值和对应的特征向量
%格式: [lambda,v,k]=mainvp(A,x,alpha,epsilon,max1)
%A为n阶方阵, x为初始向量, epsilon为上限, max1为循环次数,
%alpha为模最大的近似特征值. lambda返回按模最大的特征值,
%v返回对应的特征向量, k返回迭代次数
lambda=0; k=0; err=1;
mu=0.5; n=length(x);
A=A-alpha*eye(n);
[L,U,P]=lu(A);
while(k<max1)&(err>epsilon)
   [m,j]=max(abs(x));
   m=x(j);     y=x./m;
   z=L\(P*y);  x=U\z;
   err=abs(1/m-1/mu);
   k=k+1;  mu=m;
end
lambda=alpha+1/m;
v=y;
```

例 8.3 利用反幂法通用程序 mainvp.m, 求 A 近似于 -6.42 的特征值和对应

的特征向量, 其中

$$A=\begin{pmatrix}-1 & 2 & 1\\ 2 & -4 & 1\\ 1 & 1 & -6\end{pmatrix}.$$

解 注意到此处 $\alpha=-6.42$, 在 MATLAB 命令窗口执行:

```
>> A=[-1,2,1;2,-4,1;1,1,-6];
>> x=[1;1;1]; alpha=-6.42;
>> [lambda,v]=mainvp(A,x,alpha,1e-10,100)
lambda =
      -6.42106661430895
v =
   -0.04614548302620
   -0.37492113128274
    1.00000000000000
k =
    6
```

8.2 Jacobi 方法

Jacobi 方法用于求解实对称矩阵的全部特征值和对应的特征向量. 其数学原理如下:

(1) n 阶实对称阵的特征值全为实数, 其对应的特征向量线性无关且两两正交.

(2) 相似矩阵具有相同的特征值.

(3) 若 n 阶实矩阵 A 是对称的, 则存在正交矩阵 Q, 使得 $Q^{\mathrm{T}}AQ=D$, 其中 D 是一个对角矩阵, 它的对角元素 $\lambda_1,\lambda_2,\cdots,\lambda_n$ 就是 A 的特征值, Q 的第 i 列向量就是 λ_i 对应的特征向量.

Jacobi 方法就是基于上述原理, 用一系列正交变换对角化 A, 即逐步消去 A 的非对角元, 从而得到 A 的全部特征值.

8.2.1 实对称矩阵的旋转正交相似变换

这里首先介绍一种正交变换, 它是 Jacobi 方法的基本工具.

定义 8.1 设 $1\leqslant i<j\leqslant n$, 则称矩阵

$$
R_{ij}=\begin{matrix}\begin{pmatrix}
1 & & & & & & & & & \\
& \ddots & & & & & & & & \\
& & 1 & & & & & & & \\
& & & \cos\varphi & & & & \sin\varphi & & \\
& & & & 1 & & & & & \\
& & & & & \ddots & & & & \\
& & & & & & 1 & & & \\
& & & -\sin\varphi & & & & \cos\varphi & & \\
& & & & & & & & 1 & \\
& & & & & & & & & \ddots \\
& & & & & & & & & & 1
\end{pmatrix} & \begin{matrix} \\ \\ \\ i \\ \\ \\ \\ j \\ \\ \\ \\ \end{matrix} \\ \begin{matrix} & & & i & & & & j & & & \end{matrix} & \end{matrix} \tag{8.10}
$$

为 (i,j) 平面的旋转矩阵, 或 Givens 变换矩阵.

显然, $R=R_{ij}$ 为正交矩阵, 即 $R^{\mathrm{T}}R=I$. 对于向量 $x\in\mathbf{R}^n$, 由线性变换 $y=Rx$ 得到的 y 的分量为

$$
\begin{cases}
y_i = x_i\cos\varphi + x_j\sin\varphi, \\
y_j = -x_i\sin\varphi + x_j\cos\varphi, \\
y_k = x_k, \quad k\neq i,j,
\end{cases} \tag{8.11}
$$

即用 R_{ij} 对向量 x 作用, 只改变其第 i,j 两个分量.

由矩阵 $R=R_{ij}$ 确定的正交变换 $y=Rx$ 称为平面旋转变换, 或 Givens 变换. 根据 (8.11) 容易验证, 矩阵 R_{ij} 具有下列基本性质:

定理 8.2 设 $x\in\mathbf{R}^n$ 的第 j 个分量 $x_j\neq 0$, $1\leqslant i<j\leqslant n$. 若令

$$
c=\cos\varphi=\frac{x_i}{\sqrt{x_i^2+x_j^2}}, \quad s=\sin\varphi=\frac{x_j}{\sqrt{x_i^2+x_j^2}}, \tag{8.12}
$$

则 $y=R_{ij}x$ 的分量为

$$
\begin{cases}
y_i=\sqrt{x_i^2+x_j^2}, \quad y_j=0, \\
y_k=x_k, \quad k\neq i,j.
\end{cases} \tag{8.13}
$$

上述定理表明, 可以用 Givens 变换将向量的某个分量变为零元素. 我们看下面的例子.

例 8.4 设 $x=(-2,4,-1,3)^{\mathrm{T}}$, 构造 Givens 变换 R_{24} 使得 $y=R_{24}x$ 的分量 $y_4=0$.

解 这里的 $i=2$, $j=4$. 按 (8.12), 有

$$
c=\cos\varphi=\frac{4}{\sqrt{4^2+3^2}}=\frac{4}{5}, \quad s=\sin\varphi=\frac{3}{\sqrt{4^2+3^2}}=\frac{3}{5}.
$$

由 (8.30) 得

$$\begin{pmatrix} 1 & & & \\ & \frac{4}{5} & & \frac{3}{5} \\ & & 1 & \\ & -\frac{4}{5} & & \frac{4}{5} \end{pmatrix}.$$

由于 $y_2=\sqrt{4^2+3^2}=5$, 故由 (8.13) 得知: $y=R_{24}x=(-2,5,-1,0)^{\mathrm{T}}$.

下面我们来讨论 Givens 变换对实对称矩阵的作用. 用旋转矩阵 R_{ij} 对实对称矩阵 $A=(a_{ij})_{n\times n}$ 作正交相似变换, 所得矩阵记为 A_1, 即

$$A_1=R_{ij}AR_{ij}^{\mathrm{T}}=(a_{ij}^{(1)}).$$

显然

$$A_1^{\mathrm{T}}=(R_{ij}AR_{ij}^{\mathrm{T}})^{\mathrm{T}}=R_{ij}AR_{ij}^{\mathrm{T}}=A_1,$$

即 A_1 仍为实对称矩阵. 直接计算可得

$$\begin{aligned}
a_{ii}^{(1)}&=a_{ii}\cos^2\varphi+a_{jj}\sin^2\varphi+2a_{ij}\cos\varphi\sin\varphi,\\
a_{jj}^{(1)}&=a_{ii}\sin^2\varphi+a_{jj}\cos^2\varphi-2a_{ij}\cos\varphi\sin\varphi,\\
a_{il}^{(1)}&=a_{li}^{(1)}=a_{il}\cos\varphi+a_{jl}\sin\varphi,\quad l\neq i,j,\\
a_{jl}^{(1)}&=a_{lj}^{(1)}=-a_{il}\sin\varphi+a_{jl}\cos\varphi,\quad l\neq i,j,\\
a_{lm}^{(1)}&=a_{ml}^{(1)}=a_{ml},\quad m,l\neq i,j,\\
a_{ij}^{(1)}&=a_{ji}^{(1)}=a_{ij}(\cos^2\varphi-\sin^2\varphi)-(a_{ii}-a_{jj})\cos\varphi\sin\varphi.
\end{aligned}\tag{8.14}$$

不难看出, A 经过 R_{ij} 的正交相似变换后, A_1 的元素和 A 的元素相比, 只有第 i 行, 第 j 行和第 i 列, 第 j 列元素发生了变化, 而其它元素和 A 是相同的.

由 (8.14) 的最后一个等式可知, 若 $a_{ij}\neq 0$, 则可使适当选取 φ 的值, 使得 $a_{ij}^{(1)}=a_{ji}^{(1)}=0$. 事实上, 令

$$a_{ij}(\cos^2\varphi-\sin^2\varphi)-(a_{ii}-a_{jj})\cos\varphi\sin\varphi=0,$$

解得

$$\cot 2\varphi=\frac{a_{ii}-a_{jj}}{2a_{ij}}=\frac{1-\tan^2\varphi}{2\tan\varphi},\quad -\frac{\pi}{4}<\varphi\leqslant\frac{\pi}{4}.\tag{8.15}$$

在 Jacobi 方法中, 我们总是按上式选取 φ. 在实际计算时, 为避免使用三角函数, 可令

$$t=\tan\varphi,\quad c=\cos\varphi,\quad s=\sin\varphi,\quad d=\frac{a_{ii}-a_{jj}}{2a_{ij}}.\tag{8.16}$$

由 (8.15) 得

$$t^2 + 2dt - 1 = 0. \tag{8.17}$$

方程 (8.17) 有两个根, 取其绝对值最小者为 t, 即

$$t = \begin{cases} -d + \sqrt{d^2+1}, & d > 0, \\ 1, & d = 0, \\ -d - \sqrt{d^2+1}, & d < 0. \end{cases} \tag{8.18}$$

若记

$$c = \cos\varphi = \frac{1}{\sqrt{1+t^2}}, \quad s = \sin\varphi = \frac{t}{\sqrt{1+t^2}}, \tag{8.19}$$

这时, (8.14) 可写为

$$\begin{aligned} &a_{ii}^{(1)} = a_{ii}c^2 + a_{jj}s^2 + 2csa_{ij}, \\ &a_{jj}^{(1)} = a_{ii}s^2 + a_{jj}c^2 - 2csa_{ij}, \\ &a_{il}^{(1)} = a_{li}^{(1)} = ca_{il} + sa_{jl}, \quad l \neq i, j, \\ &a_{jl}^{(1)} = a_{lj}^{(1)} = -sa_{il} + ca_{jl}, \quad l \neq i, j, \\ &a_{lm}^{(1)} = a_{ml}^{(1)} = a_{ml}, \quad m, l \neq i, j, \\ &a_{ij}^{(1)} = a_{ji}^{(1)} = 0. \end{aligned} \tag{8.20}$$

8.2.2 Jacobi 方法

选择 $A_0 = A$ 中一对非零的非对角元素 a_{ij}, a_{ji}, 使用平面旋转矩阵 R_{ij} 作正交相似变换得 A_1, 可使 A_1 的这对非对角元素 $a_{ij}^{(1)} = a_{ji}^{(1)} = 0$; 再选择 A_1 中一对非零的非对角元素作上述旋转正交相似变换得 A_2, 可使 A_2 的这对非对角元素为 0. 如此不断地做旋转正交相似变换, 可产生一个矩阵序列 $A = A_0, A_1, \cdots, A_k, \cdots$. 虽然 A 至多只有 $n(n-1)/2$ 对非零非对角元素, 但不能期望通过 $n(n-1)/2$ 次旋转正交相似变换使其对角化. 因为每次旋转变换虽然能使一对特定的非对角元素化为 0, 但这次变换可能将前面已经化为 0 的一对非对角元素变成非 0.

但是, 在 Jacobi 方法中的每一步, 比如由 A_{k-1} 变成 A_k, 取其绝对值最大的一对非零非对角元素, 即取

$$\left|a_{i_k j_k}^{(k-1)}\right| = \max_{\substack{1 \leqslant i,j \leqslant n \\ i \neq j}} \left|a_{ij}^{(k-1)}\right| \tag{8.21}$$

作旋转相似变换, 这时记旋转矩阵 $R_{ij} = R_{i_k j_k}$. 后面将证明, 这样产生的矩阵序列 $A_0, A_1, \cdots, A_k, \cdots$ 趋向于对角矩阵, 即 Jacobi 方法是收敛的.

在实际计算中, 可预先取一个小的控制量 $\varepsilon > 0$, 若成立

$$\left|a_{ij}^{(k)}\right| < \varepsilon, \quad i,j=1,2,\cdots,n,\ i \neq j, \tag{8.22}$$

则可视 A_k 为对角矩阵, 从而结束计算. A_k 的对角元素可视为 A 的特征值.

Jacobi 方法也可以求 A 的所有特征向量. 事实上, 由

$$\begin{aligned} A_k &= R_k A_{k-1} R_k^{\mathrm{T}} = R_k R_{k-1} A_{k-2} R_{k-1}^{\mathrm{T}} R_k^{\mathrm{T}} = \cdots \\ &= R_k R_{k-1} \cdots R_1 A R_1^{\mathrm{T}} \cdots R_{k-1}^{\mathrm{T}} R_k^{\mathrm{T}}, \end{aligned}$$

若记

$$Q_k = R_1^{\mathrm{T}} \cdots R_{k-1}^{\mathrm{T}} R_k^{\mathrm{T}}, \tag{8.23}$$

则

$$A_k = Q_k^{\mathrm{T}} A Q_k. \tag{8.24}$$

这里, Q_k 为正交矩阵. 若 A_k 可视为对角矩阵, 其对角元即为 A 的特征值, 其第 i 个对角元 $a_{ii}^{(k)}$ 对应的特征向量就是 Q_k 的第 i 列元素构成的向量. Q_k 的计算可与 A 的旋转相似变换同步进行. 若令 $Q_0 = I$, 则

$$Q_k = Q_{k-1} R_k^{\mathrm{T}}. \tag{8.25}$$

若 $R_k = R_{ij}$, 得 Q_k 的计算公式如下:

$$\begin{cases} q_{li}^{(k)} = q_{li}^{(k-1)} c + q_{lj}^{(k-1)} s, & l = 1,2,\cdots,n, \\ q_{lj}^{(k)} = -q_{li}^{(k-1)} s + q_{lj}^{(k-1)} c, & l = 1,2,\cdots,n, \\ q_{km}^{(k)} = q_{km}^{(k-1)}, & k,m \neq i,j. \end{cases} \tag{8.26}$$

也就是说, 除了第 i,j 列元素发生变化外, 其它元素不变. 若不需要计算特征向量, 则可省略此步.

根据上讨论, 可得 Jacobi 方法的计算步骤如下:

算法 8.3 (Jacobi 方法)

步 1 输入矩阵 A, $Q = I$, 初始向量 x, 误差限 ε, 最大迭代次数 N, 置 $k := 1$;

步 2 在矩阵中找绝对值最大的非对角元

$$\mu = \left|a_{i_r j_r}\right| = \max_{\substack{1 \leqslant i,j \leqslant n \\ i \neq j}} |a_{ij}|,$$

置 $i := i_r$, $j := j_r$;

步 3 按 (8.16) ~ (8.20) 式计算 d,t,c,s 的值和矩阵 A_1 的元素 $a_{lm}^{(1)}$, $l,m = 1,2,\cdots,n$;

步 4 更新 Q 的元素:

$$\begin{cases} q_{li} := q_{li}c + q_{lj}s, \\ q_{lj} := -q_{li}s + q_{lj}c, \end{cases} \quad l = 1, 2, \cdots, n.$$

步 5 若 $\mu < \varepsilon$, 输出 A_1 的对角元和 Q 的列向量, 停算；否则, 转步 6.

步 6 若 $k < N$, 置 $k := k + 1$, 转步 2; 否则输出计算失败信息, 停算.

我们给出 Jacobi 迭代法的 MATLAB 通用程序如下:

```
%majacobieig.m
function [V,D]=majacobieig(A,tol)
%用途: 用Jacobi迭代法求实对称矩阵A的特征值和特征向量
%格式: [V,D]=magauss(A,tol), A为n阶对称方阵, tol为容许误差,
%V是特征向量矩阵, D是n阶对角矩阵, 其对角元为矩阵A的n个特征值
if nargin<2,tol=1e-3;end
[n,n]=size(A);
%初始化
D=A; V=eye(n); flag=1;
%计算A的非对角元绝对值最大元素所在的行t和列r
[w1,p]=max(abs(D-diag(diag(D))));
[w2,q]=max(w1); p=p(q);
while(flag==1)
  d=(D(q,q)-D(p,p))/(2*D(p,q));
  %t=sign(d)/(abs(d)+sqrt(1+d^2));
  if(d>0)
     t=-d+sqrt(d^2+1);
  else if(d<0)
      t=-d-sqrt(d^2+1);
    else
      t=1;
    end
  end
  c=1/sqrt(t^2+1);  s=c*t;
  R=[c s; -s c];
  D([p q],:)=R'*D([p q],:);
  D(:,[p q])=D(:,[p q])*R;
  V(:,[p q])=V(:,[p q])*R;
```

```
  [w1,p]=max(abs(D-diag(diag(D))));
  [w2,q]=max(w1); p=p(q);
  if (abs(D(p,q))<tol*sqrt(sum(diag(D).^2)/n))
    flag=0;
  end
end
D=diag(diag(D));
```

例 8.5 **利用 Jacobi 迭代法通用程序 majacobieig.m, 求实对称矩阵 A 的全部特征值和对应的特征向量, 其中**

$$A=\begin{pmatrix} 3 & 2 & 1 \\ 2 & 5 & 4 \\ 1 & 4 & 7 \end{pmatrix}.$$

解 在 MATLAB 命令窗口执行:

```
>> A=[3 2 1; 2 5 4; 1 4 7];
>> [V,D]=majacobieig(A,1e-5)
V =
    0.7765   -0.5749    0.2580
    0.3258    0.7167    0.6166
   -0.5394   -0.3947    0.7438
D =
    3.1444         0         0
         0    1.1930         0
         0         0   10.6625
```

8.2.3 Jacobi 方法的收敛性

我们现在来证明 Jacobi 方法的收敛性. 记实对称矩阵 A 的非对角元素的平方和为

$$S(A)=\sum_{\substack{l,m=1\\ l\neq m}}^{n} a_{lm}^2. \tag{8.27}$$

设 $A_{k+1}=R_{ij}A_kR_{ij}^{\mathrm{T}}$, 则由 (8.20) 不难验证

$$S(A_{k+1})=S(A_k)-2\big[a_{ij}^{(k)}\big]^2. \tag{8.28}$$

即经过这种正交相似变换后, A_{k+1} 的非对角元素平方和减少了 $2[a_{ij}^{(k)}]^2$. 同时, 容易验证

$$\big[a_{ii}^{(k+1)}\big]^2+\big[a_{jj}^{(k+1)}\big]^2=\big[a_{ii}^{(k)}\big]^2+\big[a_{jj}^{(k)}\big]^2+2\big[a_{ij}^{(k)}\big]^2, \tag{8.29}$$

即对角元素的平方和增加了 $2[a_{ij}^{(k)}]^2$. 若在 Jacobi 方法中, 每次旋转正交相似变换使 A_k 的绝对值最大的非对角元素化为 0, 则成立以下定理:

定理 8.3 记实对称矩阵 $A=A_0$, 若在 Jacobi 方法中, 每次旋转正交相似变换使 A_k 的绝对值最大的非对角元素化为 0, 则得到的矩阵序列 $\{A_k\}$ 趋向于对角矩阵.

证 设 A_k 的绝对值最大的非对角元素为 $a_{ij}^{(k)}$, 故有

$$[a_{ij}^{(k)}]^2 \geqslant \frac{1}{n(n-1)}S(A_k).$$

用旋转正交相似变换将其化为 0, 得 A_{k+1}, 此时

$$\begin{aligned} S(A_{k+1}) &= S(A_k) - 2[a_{ij}^{(k)}]^2 \leqslant S(A_k) - \frac{2}{n(n-1)}S(A_k) \\ &= \left[1-\frac{2}{n(n-1)}\right]S(A_k) \leqslant \left[1-\frac{2}{n(n-1)}\right]^{k+1}S(A). \end{aligned}$$

由于

$$1-\frac{2}{n(n-1)}<1,$$

所以

$$\lim_{k\to\infty} S(A_{k+1}) = 0,$$

即 A_{k+1} 趋向于对角矩阵, 故 Jacobi 方法是收敛的. □

8.3 QR 方法

QR 方法用于求一般矩阵的全部特征值, 是目前最有效的方法之一. 本节就实矩阵的情形进行介绍.

8.3.1 Householder 变换

定义 8.2 设 v 是 $n\times 1$ 向量, 满足 $v^{\mathrm{T}}v=1$, 令 $H=I-2vv^{\mathrm{T}}$, 则称 H 为 Householder 矩阵, 又称为初等反射阵.

根据上述定义, 容易看出

$$H^{\mathrm{T}} = (I-2vv^{\mathrm{T}})^{\mathrm{T}} = I-2vv^{\mathrm{T}} = H,$$

且

$$HH^{\mathrm{T}} = HH = (I-2vv^{\mathrm{T}})(I-2vv^{\mathrm{T}}) = I-4vv^{\mathrm{T}}+4v(v^{\mathrm{T}}v)v^{\mathrm{T}} = I,$$

故 H 是对称正交阵.

定理 8.4 设 x, y 是两个不相等的 n 维列向量，且满足 $\|x\|_2 = \|y\|_2$，则存在一个 Householder 矩阵 H，使得 $Hx = y$.

证 令 $v = (x-y)/\|x-y\|_2$，则有 Householder 矩阵

$$H = I - 2vv^{\mathrm{T}} = I - 2\frac{(x-y)(x^{\mathrm{T}} - y^{\mathrm{T}})}{\|x-y\|_2^2}.$$

于是

$$Hx = x - 2\frac{(x-y)(x^{\mathrm{T}} - y^{\mathrm{T}})}{\|x-y\|_2^2}x = x - 2\frac{(x-y)(x^{\mathrm{T}}x - y^{\mathrm{T}}x)}{\|x-y\|_2^2},$$

注意到 $\|x-y\|_2^2 = (x-y)^{\mathrm{T}}(x-y) = 2(x^{\mathrm{T}}x - y^{\mathrm{T}}x)$，故 $Hx = x - (x-y) = y$. □

推论 8.1 设 x 是 n 维列向量，$a = \pm\|x\|_2$，且 $x \neq -ae_1$，则存在一个 Householder 矩阵

$$H = I - 2\frac{uu^{\mathrm{T}}}{\|u\|_2^2} = I - \rho^{-1}uu^{\mathrm{T}}, \tag{8.30}$$

使得

$$Hx = -ae_1,$$

其中 $e_1 = (1, 0, \cdots, 0)^{\mathrm{T}}$，$u = x + ae_1$，$\rho = \|u\|_2^2/2$.

下面讨论推论 8.1 中参数 a 符号的取法. 设 $x = (x_1, x_2, \cdots, x_n)^{\mathrm{T}} \neq 0$，$u = (u_1, u_2, \cdots, u_n)^{\mathrm{T}}$，则

$$u = x + ae_1 = (x_1 + a, x_2, \cdots, x_n)^{\mathrm{T}},$$

$$\rho = \frac{1}{2}\|u\|_2^2 = \frac{1}{2}\left[(x_1+a)^2 + x_2^2 + \cdots + x_n^2\right] = a(a + x_1).$$

由上式可以看出，如果 a 与 x_1 异号，则计算 $x_1 + a$ 时有效数字可能会损失，故取 a 与 x_1 有相同的符号，即取 $a = \mathrm{sgn}(x_1)\|x\|_2$，其中

$$\mathrm{sgn}(x) = \begin{cases} 1, & x > 0, \\ 0, & x = 0, \\ -1, & x < 0. \end{cases}$$

下面给出 Householder 矩阵变换的 MATLAB 通用程序.

- Householder 矩阵变换的 MATLAB 程序

```
%househ.m
function H=househ(x)
%用途：对于向量x,构造Householder变换矩阵H,使得Hx=(*,0,...,0)'
%格式：function H=househ(x)  x为输入列向量,
%H返回Householder变换矩阵
```

```
n=length(x);
I=diag(ones(1,n));
sn=sign(x(1));
if sn==0  sn=1; end
z=x(2:n);
if norm(z,inf)==0
   H=I; return;
end
a=sn*norm(x,2);
u=x;
u(1)=u(1)+a;
rho=a*(a+x(1));
H=I-1.0/rho*u*u';
```

例 8.6 利用 Householder 变换通用程序 househ.m, 将列向量 $x=(2,1,-3,4)^{\mathrm{T}}$ 的后三个分量化为零.

解 在 MATLAB 命令窗口执行:

```
>> x=[2 1 -3 4]';
>> H=househ(x); H*x
ans =
     -5.4772
           0
           0
           0
```

8.3.2 化一般矩阵为拟上三角矩阵

在用 QR 方法求矩阵特征值时, Householder 矩阵有两个作用: 一是对 A 作正交相似变换, 把 A 化为拟上三角矩阵; 二是对矩阵作正交三角分解. 其中拟上三角阵是指次对角线以下的元素全为零的矩阵, 即

我们首先讨论把 A 化为拟上三角矩阵. 设 $A_1=A=(a_{ij}^{(1)})$ 是 n 阶实方阵, 取 $x=(0,a_{21}^{(1)},\cdots,a_{n1})^{\mathrm{T}}$, 记 $a_1=a=\mathrm{sgn}(x_2)\|x\|_2$, 则由推论 8.1 构造 Householder 矩

阵

$$H_1 = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & * & \cdots & * \\ \vdots & \vdots & & \vdots \\ 0 & * & \cdots & * \end{pmatrix},$$

使得

$$H_1 x = a_1 e_2.$$

所以 $H_1 A_1$ 的第 1 列为

$$H_1 \begin{pmatrix} a_{11}^{(1)} \\ a_{21}^{(1)} \\ \vdots \\ a_{n1}^{(1)} \end{pmatrix} = H_1 x + H_1 \begin{pmatrix} a_{11}^{(1)} \\ 0 \\ \vdots \\ 0 \end{pmatrix} = a_1 e_2 + \begin{pmatrix} a_{11}^{(1)} \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} a_{11}^{(1)} \\ a_1 \\ \vdots \\ 0 \end{pmatrix}.$$

因为用 H_1 右乘一个矩阵不改变该矩阵的第 1 列, 于是

$$A_2 = H_1 A_1 H_1 = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(2)} & \cdots & a_{1n}^{(2)} \\ a_1 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ 0 & a_{32}^{(2)} & \cdots & a_{3n}^{(2)} \\ \vdots & \vdots & & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} \end{pmatrix}.$$

再取 $x = (0, 0, a_{32}^{(2)}, \cdots, a_{n2}^{(2)})^{\mathrm{T}}$, 记 $a_2 = a = \mathrm{sgn}(x_3)\|x\|_2$, 构造 H_2 为

$$H_2 = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & * & \cdots & * \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & * & \cdots & * \end{pmatrix},$$

使得

$$H_2 x = a_2 e_3.$$

所以 $H_2 A_2$ 的第 1 列与 A_2 的第 1 列相同, 而 $H_2 A_2$ 的第 2 列变为

$$H_2 x + H_2 \begin{pmatrix} a_{12}^{(2)} \\ a_{22}^{(2)} \\ 0 \\ \vdots \\ 0 \end{pmatrix} = a_2 e_3 + \begin{pmatrix} a_{12}^{(2)} \\ a_{22}^{(2)} \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} a_{12}^{(2)} \\ a_{22}^{(2)} \\ a_2 \\ \vdots \\ 0 \end{pmatrix},$$

而用 H_2 右乘一个矩阵不改变该矩阵的第 1 列和第 2 列, 于是

$$A_3 = H_2A_2H_2 = \begin{pmatrix} * & * & * & \cdots & * \\ a_1 & * & * & \cdots & * \\ 0 & a_2 & * & \cdots & * \\ 0 & 0 & * & \cdots & * \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & * & \cdots & * \end{pmatrix}.$$

这样下去, 经过 $n-2$ 次变换后, A_1 就化为拟上三角阵 A_{n-1}, 即

$$A_{n-1} = H_{n-2}\cdots H_2H_1A_1H_1H_2\cdots H_{n-2}$$
$$= \begin{pmatrix} * & * & * & * & \cdots & * \\ a_1 & * & * & * & \cdots & * \\ & a_2 & * & * & \cdots & * \\ & & a_3 & * & \cdots & * \\ & & & \ddots & \ddots & \vdots \\ & & & & a_{n-1} & * \end{pmatrix}.$$

如果 A_1 是对称矩阵, 则 A_{n-1} 仍是对称矩阵, 此时 A_{n-1} 将是对称三对角矩阵:

$$A_{n-1} = \begin{pmatrix} * & a_1 & & & & \\ a_1 & * & a_2 & & & \\ & a_2 & * & a_3 & & \\ & & a_3 & * & \ddots & \\ & & & \ddots & \ddots & a_{n-1} \\ & & & & a_{n-1} & * \end{pmatrix}.$$

下面给出将矩阵 A 化为拟上三角矩阵的 MATLAB 通用程序.

- 化矩阵 A 为拟上三角矩阵的 MATLAB 程序

```
%hessen.m
function A=hessen(A)
%用途: 用Householder变换化矩阵A为拟上三角阵.
%输入: n阶实方阵A
%输出: A的Hessenberg形
%调用函数: househ.m
[n,n]=size(A);
for k=1:(n-2)
    x=A(k+1:n,k); H=househ(x);
    A(k+1:n,1:n)=H*A(k+1:n,1:n);
    A(1:n,k+1:n)=A(1:n,k+1:n)*H;
end
```

例 8.7 利用通用程序 hessen.m, 将下列矩阵化为拟上三角矩阵:

$$A=\begin{pmatrix}1&2&3&4\\3&4&1&2\\4&1&2&3\\2&3&4&1\end{pmatrix}.$$

解 在 MATLAB 命令窗口执行:

```
>> A=[1 2 3 4; 3 4 1 2; 4 1 2 3; 2 3 4 1];
>> A=hessen(A)
A =
     1.0000   -4.8281    2.1574    1.0175
    -5.3852    6.2759   -1.7979    0.5586
    -0.0000   -3.0838   -1.4804   -0.9560
    -0.0000   -0.0000   -0.7703    2.2046
```

8.3.3 矩阵的正交三角分解

设 $A^{(1)}=A=(a_{ij}^{(1)})$ 是 n 阶实方阵, 取 $x=(a_{11}^{(1)},a_{21}^{(1)},\cdots,a_{n1}^{(1)})^{\mathrm{T}}$, 记

$$a_1=a=\operatorname{sgn}(x_1)\|x\|_2,$$

构造 Householder 矩阵 H_1, 则

$$A^{(2)}=H_1A^{(1)}=\begin{pmatrix}a_1&a_{12}^{(2)}&\cdots&a_{1n}^{(2)}\\0&a_{22}^{(2)}&\cdots&a_{2n}^{(2)}\\\vdots&\vdots&&\vdots\\0&a_{n2}^{(2)}&\cdots&a_{nn}^{(2)}\end{pmatrix}.$$

再取 $x=(0,a_{22}^{(2)},\cdots,a_{n2}^{(2)})^{\mathrm{T}}$, 记 $a_2=a=\operatorname{sgn}(x_2)\|x\|_2$, 构造 Householder 矩阵 H_2, 则

$$A^{(3)}=H_2A^{(2)}=\begin{pmatrix}a_1&a_{12}^{(2)}&a_{13}^{(2)}&\cdots&a_{1n}^{(2)}\\0&a_2&a_{23}^{(3)}&\cdots&a_{2n}^{(3)}\\0&0&a_{33}^{(3)}&\cdots&a_{3n}^{(3)}\\\vdots&\vdots&\vdots&&\vdots\\0&0&a_{n3}^{(3)}&\cdots&a_{nn}^{(3)}\end{pmatrix}.$$

经过 $n-1$ 次变换后, $A^{(1)}$ 被化为上三角阵 $A^{(n)}$:

$$A^{(n)}=H_{n-1}H_{n-2}\cdots H_1A^{(1)}=\begin{pmatrix}a_1&*&*&\cdots&*\\&a_2&*&\cdots&*\\&&a_3&\cdots&*\\&&&\ddots&\vdots\\&&&&a_n\end{pmatrix}$$

记 $Q = H_1H_2\cdots H_{n-1}$, $R = A^{(n)}$, 则有

$$A = H_1^{-1}H_2^{-1}\cdots H_{n-1}^{-1}R = H_1H_2\cdots H_{n-1}R = QR.$$

因为 Q 是正交矩阵 $H_i\,(i = 1,2,\cdots,n-1)$ 的乘积, 故也是正交矩阵, R 是上三角矩阵, 这种分解称为正交三角分解, 也叫 QR 分解.

8.3.4 基本 QR 方法及其通用程序

现在我们来介绍求一般方阵全部特征值的 QR 方法. 令 $A_1 = A$, 对 A_1 作 QR 分解:

$$A_1 = Q_1R_1,$$

然后令 $A_2 = R_1Q_1$, 再对 A_2 作 QR 分解:

$$A_2 = Q_2R_2,$$

并令 $A_3 = R_2Q_2$, 这样下去就得到一个矩阵序列 $\{A_k\}$, 其产生过程可概述如下:

$$\begin{cases} A_1 = A, \\ A_k = Q_kR_k, \\ A_{k+1} = R_kQ_k \end{cases} \quad (k = 1,2,\cdots). \tag{8.31}$$

容易证明, A_{k+1} 与 A_k 相似, 故 $\{A_k\}$ 有相同的特征值.

在一定条件下, $\{A_k\}$ 本质上收敛于上三角矩阵 (或分块上三角阵). 若它们收敛于上三角阵, 则该上三角阵的对角元就是原矩阵 A 的全部特征值; 若收敛于分块上三角阵, 则这些分块矩阵的特征值值也就是 A 的特征值.

由于当 A 为一般的实矩阵时, $\{A_k\}$ 的收敛速度较慢, 故在 QR 方法的实际应用中, 通常先将 A 化为相似的拟上三角阵, 再求特征值以加快收敛速度. 它的计算过程如下:

算法 8.4 (基本 QR 方法)

步 1 输入矩阵 $A \in \mathbf{R}^{n\times n}$;

步 2 初始化: A_1 为 A 的拟上三角形矩阵;

步 3 迭代过程: 对于 $k = 1,2,\cdots$

(1) $A_k = Q_kR_k$ (QR 分解),

(2) $A_{k+1} = Q_k^{\mathrm{T}}A_kQ_k = R_kQ_k$ (正交相似变换).

下面给出基本 QR 方法的 MATLAB 通用程序.

- 基本 QR 方法 MATLAB 程序

```
%qralg.m
function [iter,D]=qralg(A)
%用途: 用基本QR算法求实方阵的全部特征值.
%输入: n阶实方阵A
%输出: 迭代次数iter, A全部特征值D
%调用函数: hessen.m, qrtran.m, eig--仅用于1,2矩阵
ep=0.5*1e-4; [n,n]=size(A);
D=zeros(1,n); i=n; m=n; iter=0;  %初始化
A=hessen(A);  %化矩阵A为Hessenbeg形
%用基本QR算法进行迭代
while(n>0)
   if m<=2
      la=eig(A(1:m,1:m)); D(1:m)=la';  break;
   end
   iter=iter+1;
   A=qrtran(m,A); %对A的左上角的m阶对角块作QR变换
   for k=m-1:-1:1
      if abs(A(k+1,k))<ep
         if m-k<=2
            la=eig(A(k+1:m,k+1:m));
            j=i-m+k+1; D(j:i)=la';
            i=j-1; m=k; break;
         end
      end
   end
end
%qrtran.m
function A=qrtran(m,A)
%功能: 对矩阵A的左上角的m阶对角块作QR变换: 先用Givens变换作
%QR分解A=QR,再作相似变换A:=Q'AQ=RQ.
%输入: n阶HessenbergA, 其中A(m+1,m)=0,m>2.
%输出: 变换后的Hessenberg形矩阵A.
Q=diag(ones(1,m));
for i=1:m-1
   xi=A(i,i);  xk=A(i+1,i);
```

```
        if xk~=0
           d=sqrt(xi^2+xk^2); c=xi/d;
           s=xk/d; J=[c, s;-s,c];
           A(i:i+1,i:m)=J*A(i:i+1,i:m);
           Q(1:m,i:i+1)=Q(1:m,i:i+1)*J';
        end
    end
    A(1:m,1:m)=A(1:m,1:m)*Q;
```

例 8.8 利用通用程序 qralg.m, 求下列矩阵的全部特征值:

$$A=\begin{pmatrix} 3 & 2 & 3 & 4 & 5 & 6 & 7 \\ 11 & 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 8 & 9 & 1 & 2 & 3 & 4 \\ -4 & 2 & 9 & 11 & 13 & 15 & 8 \\ -1 & -2 & -3 & -1 & -1 & -1 & -1 \\ 3 & 2 & 3 & 4 & 13 & 15 & 8 \\ -2 & -2 & -3 & -4 & -5 & -3 & -3 \end{pmatrix}.$$

解 在 MATLAB 命令窗口执行:

```
>> A=[3 2 3 4 5 6 7;11 1 2 3 4 5 6;2 8 9 1 2 3 4; -4 2 9 11 13 15 8;
     -1 -2 -3 -1 -1 -1 -1; 3 2 3 4 13 15 8; -2 -2 -3 -4 -5 -3 -3];
>> [iter,D]=qralg(A)
iter =
     537
D =
   18.4123    11.1805    1.7100-4.2522i   1.7100+4.2522i
   4.4982    -2.2327    -0.2783
```

习 题 8

(I) 理论分析题

8.1 用幂法求矩阵

$$A=\begin{pmatrix} 1 & -1 & 2 \\ -2 & 0 & 5 \\ 6 & -3 & 6 \end{pmatrix}$$

按模最大的特征值和相应的特征向量.

8.2 用幂法求矩阵

$$A=\begin{pmatrix}1 & -3 & 2\\ 4 & 4 & -1\\ 6 & 3 & 5\end{pmatrix}$$

按模最大的特征值和相应的特征向量.

8.3 已知矩阵

$$A=\begin{pmatrix}2 & -1 & 0\\ -1 & 2 & -1\\ 0 & -1 & 2\end{pmatrix}$$

有一个近似特征值 $\bar{\lambda}=3.41$, 用反幂法求对应的特征向量, 并改进特征值的精度.

8.4 用 Jacobi 方法求矩阵

$$A=\begin{pmatrix}1 & 2 & 0\\ 2 & -1 & 1\\ 0 & 1 & 3\end{pmatrix}$$

的全部特征值与特征向量.

8.5 用 Jacobi 方法求矩阵

$$A=\begin{pmatrix}2 & -1 & 0\\ -1 & 3 & -1\\ 0 & -1 & 5\end{pmatrix}$$

的全部特征值与特征向量.

8.6 用镜面反射变换求矩阵

$$A=\begin{pmatrix}1 & 1 & 1\\ 2 & -1 & -1\\ 2 & -4 & 5\end{pmatrix}$$

的 QR 分解.

8.7 设 A 为 $n\times n$ 实对称矩阵, 其特征值排序为 $\lambda_1\geqslant\lambda_2\geqslant\cdots\geqslant\lambda_n$, 对应的特征向量 $x_1,x_2,\cdots,x_n$ 组成标准正交组, 即 $(x_i,x_j)=\delta_{ij}$, 证明:

(1) $\lambda_n\leqslant\dfrac{(Ax,x)}{(x,x)}\leqslant\lambda_1$, 对 $\forall x\neq 0, x\in\mathbf{R}^n$;

(2) $\lambda_1=\max\limits_{x\neq 0}\dfrac{(Ax,x)}{(x,x)},\quad \lambda_n=\min\limits_{x\neq 0}\dfrac{(Ax,x)}{(x,x)}$.

8.8 设 A 为实对称矩阵, $\{A_k\}$ 是按算法 8.3 (Jacobi 方法) 产生的矩阵序列, 记

$$S(A_k)=\sum_{\substack{i,j=1\\ i\neq j}}^{n}\left[a_{ij}^{(k)}\right]^2,$$

证明：

$$\lim_{k\to\infty} S(A_k) = 0.$$

8.9 设 A 为 $n\times n$ 非奇异实矩阵，其 QR 分解为 $A=QR$. 记 $\bar{A}=RQ$, 证明：

(1) 若 A 对称，则 $\bar{A}$ 也对称；

(2) 若 A 是拟上三角矩阵，则 $\bar{A}$ 也是拟上三角矩阵.

8.10 设

$$A=\begin{pmatrix} (A_{11})_{3\times3} & O_{3\times2} \\ O_{2\times3} & (A_{22})_{2\times2} \end{pmatrix}.$$

又设 λ_i 为 A_{11} 的特征值，λ_j 为 A_{22} 的特征值，$x_i=(\alpha_1,\alpha_2,\alpha_3)^{\mathrm{T}}$ 是 A_{11} 对应于 λ_i 的特征向量，$y_j=(\beta_1,\beta_2)^{\mathrm{T}}$ 是 A_{22} 对应于 λ_j 的特征向量. 求证：

(1) λ_i, λ_j 为 A 的特征向量；

(2) $\bar{x}_i=(\alpha_1,\alpha_2,\alpha_3,0,0)^{\mathrm{T}}$ 是 A 对应于 λ_i 的特征向量，$\bar{y}_j=(0,0,0,\beta_1,\beta_2)^{\mathrm{T}}$ 是 A 对应于 λ_j 的特征向量.

（Ⅱ）上机实验题

8.1 编制幂法的 MATLAB 程序，计算下列矩阵按模最大的特征值和相应的特征向量：

(1) $A=\begin{pmatrix} 2 & 0 & 0 \\ 2 & -1 & 1 \\ 1 & 1 & -1 \end{pmatrix}$； (2) $B=\begin{pmatrix} -3 & 1 & 0 \\ 1 & -3 & -3 \\ 0 & -3 & 4 \end{pmatrix}$.

8.2 已知矩阵

$$A=\begin{pmatrix} -1 & 2 & 0 \\ 2 & -4 & 1 \\ 1 & 1 & -6 \end{pmatrix}$$

有一个近似的特征值 $\lambda\approx-6.42$, 用反幂法编制 MATLAB 程序计算相应的特征向量，并改进特征值的精度.

8.3 用 Jacobi 方法编制 MATLAB 程序，计算下列矩阵的全部特征值：

(1) $A=\begin{pmatrix} 3 & -2 & -4 \\ -2 & 6 & -2 \\ -4 & -2 & 3 \end{pmatrix}$； (2) $B=\begin{pmatrix} 4 & -1 & & \\ -1 & 4 & -1 & \\ & -1 & 4 & -1 \\ & & -1 & 4 \end{pmatrix}$.

8.4 用基本 QR 方法编制 MATLAB 程序，计算下列矩阵的全部特征值：

(1) $A=\begin{pmatrix} -4 & 9 & 16 \\ 2 & -2 & -4 \\ -4 & 7 & 12 \end{pmatrix}$； (2) $B=\begin{pmatrix} 0 & 1 & 2 & 2 \\ 2 & 3 & 0 & 1 \\ 3 & 0 & 1 & 2 \\ 1 & 2 & 3 & 0 \end{pmatrix}$.

附录一　数值实验

数值实验是“数值分析”或“计算方法”课程中不可缺少的组成部分. 通过对典型算法的数值实验, 能有效地回顾相应章节的主要内容, 加深对实验所涉及的基本理论和方法的理解, 以及对相关数值算法的优缺点和使用范围的进一步了解.

每个数值实验都应该以实验报告的形式完成. 每个实验者应以认真的态度来对待每个实验, 实验之前要进行必要的实验准备工作, 并选用一种计算机语言 (推荐使用数学软件 MATLAB), 独立完成算法的程序编制和调试, 并在计算机上实现或演示实验结果. 值得指出的是, 实验结果分是数值实验的重要环节, 获得实验结果, 并不意味着该实验的结束, 还需要对实验结果进行认真的分析. 只有这样, 才能对实验的目的和方法获得进一步的理解, 才能对该实验的重要性获得全面而充分的认识.

A.1　数值实验报告的格式

做数值实验, 做好实验报告是必要的. 每个数值实验都应该以实验报告的形式完成. 那么, 一个完整的数值实验报告应包括哪些内容呢? 粗略地说, 它应该由以下几个部分组成: 实验目的、实验题目、实验原理与基础理论、实验内容、实验结果和实验结果分析. 换言之, 一份完整的数值实验报告, 应该包括数据准备、基础理论、实验内容与方法, 最终要对实验的结果进行必要的分析, 以期达到对算法基本原理的感性认识, 进一步加深相关算法的效用和使用范围的全面理解.

实验报告格式如下:

实验报告

专业_______ 年级_______ 班级_______ 学号_________ 姓名_________

一、实验目的

(写清楚为什么要做这个实验, 其目的是什么, 做完这个实验要达到什么结果, 实验的注意事项是什么等.)

二、实验题目

(填写实验题目.)

三、实验原理

(将实验所涉及的基础理论、算法原理详尽列出.)

四、实验内容

(列出实验的实施方案、步骤、数据准备、算法流程图以及可能用到的实验设备 (硬件和软件).)

五、实验结果

(实验结果应包括试验的原始数据、中间结果及最终结果, 复杂的结果可以用表格或图形形式实现, 较为简单的结果可以与实验结果分析合并出现.)

六、实验结果分析

(对实验结果进行认真的分析, 进一步明确实验所涉及的算法的优缺点和使用范围. 要求实验结果应能在计算机上实现或演示, 由实验者独立编程实现, 程序清单以附录的形式给出.)

A.2 数值实验

实验一 线性方程组迭代法实验

1. 迭代法的收敛速度

实验题目: 用迭代法分别对 $n=20$, $n=200$ 解方程组 $Ax=b$, 其中

$$A=\begin{pmatrix} 4 & -\frac{1}{3} & -\frac{1}{5} & & & & & \\ -\frac{1}{3} & 4 & -\frac{1}{3} & -\frac{1}{5} & & & & \\ -\frac{1}{5} & -\frac{1}{3} & 4 & -\frac{1}{3} & -\frac{1}{5} & & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & & \\ & & -\frac{1}{5} & -\frac{1}{3} & 4 & -\frac{1}{3} & -\frac{1}{5} \\ & & & -\frac{1}{5} & -\frac{1}{3} & 4 & -\frac{1}{3} \\ & & & & -\frac{1}{5} & -\frac{1}{3} & 4 \end{pmatrix}_{n\times n}$$

(1) 选取不同的初值 x_0 和不同的右端向量 b, 给定迭代误差, 用两种迭代法计算, 观测得到的迭代向量并分析计算结果给出结论;

(2) 取定初值 x_0 和右端向量 b, 给定迭代误差, 将 A 的主对角元成倍放大, 其余元素不变, 用 Jacobi 迭代法计算多次, 比较收敛速度, 分析计算结果并给出结论.

2. SOR 迭代法松弛因子的选取

实验题目: 用逐次超松弛 (SOR) 迭代法求解方程组 $Ax = b$, 其中

$$A = \begin{pmatrix} 12 & -2 & 1 & & & & \\ -2 & 12 & -2 & 1 & & & \\ 1 & -2 & 12 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 12 & -2 & 1 \\ & & & 1 & -2 & 12 & -2 \\ & & & & 1 & -2 & 12 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{198} \\ x_{199} \\ x_{200} \end{pmatrix} = \begin{pmatrix} 5 \\ 5 \\ 5 \\ \vdots \\ 5 \\ 5 \\ 5 \end{pmatrix}.$$

(1) 给定迭代误差, 选取不同的超松弛因子 $\omega > 1$ 进行计算, 观测得到的近似解向量并分析计算结果, 给出你的结论;

(2) 给定迭代误差, 选取不同的低松弛因子 $\omega < 1$ 进行计算, 观测得到的近似解向量并分析计算结果, 给出你的结论.

实验二 线性方程组直接实验

1. 高斯消去法选主元的必要性

实验题目一: 用列主元法求解线性方程组:

$$\begin{pmatrix} 0.001 & 2.000 & 3.000 \\ -2.000 & 1.072 & 5.643 \\ -1.000 & 3.712 & 4.623 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1.000 \\ 3.000 \\ 2.000 \end{pmatrix}.$$

实验题目二: 分别用列主元法和顺序高斯消去法求解下面的线性方程组, 分析对结果的影响:

$$\begin{pmatrix} 0.3 \times 10^{-16} & 59.14 & 3 & 1 \\ 1 & 2 & 1 & 1 \\ 11.2 & 9 & 5 & 2 \\ 5.291 & -6.13 & -1 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 51.97 \\ 2 \\ 1 \\ 46.78 \end{pmatrix}.$$

2. LU 分解的优点

实验题目：给定矩阵 A 和向量 b：

$$A=\begin{pmatrix} n & n-1 & \cdots & 2 & 1 \\ & n & \cdots & 3 & 2 \\ & & \ddots & \vdots & \vdots \\ & & & n & n-1 \\ & & & & n \end{pmatrix}, \quad b=\begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}.$$

(1) 求 A 的 LU 分解, n 的值自己确定;

(2) 利用 A 的 LU 分解求解下列方程组

(a) $Ax=b$, (b) $A^2x=b$, (c) $A^3x=b$.

对方程组 (c), 若先求 $LU=A^3$, 再解 $(LU)x=b$ 有何缺点?

3. 追赶法的优点

实验题目：用追赶法分别对 $n=10$, $n=100$, $n=1000$ 解方程组 $Ax=b$, 其中

$$A=\begin{pmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 4 & -1 \\ & & & -1 & 4 \end{pmatrix}, \quad b=\begin{pmatrix} 6 \\ 5 \\ \vdots \\ 5 \\ 5 \end{pmatrix}.$$

再用 LU 分解法解此方程组, 并对二者进行比较.

实验三　插值法与拟合实验

1. 插值效果的比较

实验题目：将区间 $[-5,5]$ 10 等份, 对下列函数分别计算插值节点 x_k 的值, 进行不同类型的插值, 作出插值函数的图形并与 $y=f(x)$ 的图形进行比较:

$$f(x)=\frac{1}{1+x^2}; \qquad f(x)=\arctan x; \qquad f(x)=\frac{x^2}{1+x^4}.$$

(1) 做拉格朗日插值;

(2) 做分段线性插值;

(3) 做三次样条插值.

2. 拟合多项式实验

实验题目：给定数据点如下表所示：

x_i	−1.5	−1.0	−0.5	0.0	0.5	1.0	1.5
y_i	−4.45	−0.45	0.55	0.05	−0.44	0.54	4.55

分别对上述数据作三次多项式和五次多项式拟合，并求平方误差，作出离散函数 (x_i, y_i) 和拟合函数的图形.

实验四　数值微积分实验

1. 复化求积公式计算定积分

实验题目：用复化梯形公式、复化辛普森公式、龙贝格公式求下列定积分，要求绝对误差为 $\varepsilon = 0.5 \times 10^{-8}$，并将计算结果与精确解进行比较：

(1) $\mathrm{e}^4 = \int_1^2 \frac{2}{3} x^3 \mathrm{e}^{x^2} \mathrm{d}x$,　　　(2) $\ln 6 = \int_2^3 \frac{2x}{x^2 - 3} \mathrm{d}x$.

2. 比较一阶导数和二阶导数的数值方法

实验题目：利用等距节点的函数值和端点的导数值，用不同的方法求下列函数的一阶和二阶导数，分析各种方法的有效性，并用绘图软件绘出函数的图形，观察其特点.

(1) $y = \frac{1}{20} x^5 - \frac{11}{6} x^3, \quad x \in [0, 2]$,　　　(2) $y = \mathrm{e}^{-\frac{1}{x}}, \quad x \in [-2.5, -0.5]$.

实验五　常微分方程求解实验

1. 解初值问题各种方法比较

实验题目：给定初值问题

$$\begin{cases} \frac{\mathrm{d}y}{\mathrm{d}x} = \frac{y}{x} + x\mathrm{e}^2, \quad 1 < x \leqslant 2, \\ y(1) = 0, \end{cases}$$

其精确解为 $y = x(\mathrm{e}^x - \mathrm{e})$，按

(1) 欧拉法，步长 $h = 0.025$, $h = 0.1$;

(2) 改进欧拉法，步长 $h = 0.05$, $h = 0.01$;

(3) 四阶标准龙格–库塔法，步长 $h = 0.1$;

求在节点 $x_k = 1 + 0.1k\,(k = 1, 2, \cdots, 10)$ 处的数值解及误差，比较个方法的优缺点.

2. 常微分方程性态和龙格–库塔法稳定性

实验题目：给定常微分方程初值问题

$$\begin{cases} \dfrac{\mathrm{d}y}{\mathrm{d}x} = \lambda y - \lambda x + 1, \quad 0 < x < 1, \\ y(0) = 1, \end{cases}$$

其中 $-50 \leqslant \lambda \leqslant 50$.

要求：(1) 对参数 λ 取不同的值, 取步长 $h = 0.01$, 用四阶经典龙格–库塔法计算, 将计算结果画图比较, 并分析相应的初值问题的性态;

(2) 取参数 λ 为一个绝对值较小的负数和两个不同的步长 h, 一个步长使 λ, h 在经典龙格–库塔法的稳定域内, 另一个在稳定域外, 分别用龙格–库塔法计算并比较计算结果, 取全域等距的 10 个点上的计算值.

3. 刚性方程计算

实验题目：给定刚性微分方程

$$\begin{cases} \dfrac{\mathrm{d}y}{\mathrm{d}x} = -600y + 1199.8\mathrm{e}^{-0.1x} - 600, \quad 0 < x \leqslant 5, \\ y(00) = 2, \end{cases}$$

其精确解为 $y(x) = \mathrm{e}^{-600x} + 2\mathrm{e}^{-0.1x} - 1$. 任选一显式方法, 取不同的步长求解, 并分析计算结果.

实验六 非线性方程求根实验

1. 迭代函数对收敛性的影响

实验题目：用简单迭代法求方程 $f(x) = 3x^3 - 4x + 1 = 0$ 的根.

方案一 化方程为等价不动点方程

$$x = \sqrt[3]{\frac{4x-1}{3}} = \varphi(x),$$

取初值 $x_0 = 0.5$, 迭代 8 次.

方案二 化方程为等价方程

$$x = \frac{3x^3+1}{4} = \varphi(x),$$

取初值 $x_0 = 0.5$, 迭代 8 次, 观察其计算结果, 并加以分析.

2. 初值的选择对收敛性的影响

实验题目：用牛顿法求方程 $f(x) = x^3 + x - 1 = 0$ 在 $x = 0.5$ 附近的根.

方案一 使用牛顿法并取初值 $x_0 = 0.5$, 由

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

得

$$x_{k+1} = x_k - \frac{x_k^3 + x_k - 1}{3x_k^2 + 1} = \frac{2x_k^3 + 1}{3x_k^2 + 1},$$

迭代 6 次.

方案二 取初值 $x_0 = 0.0$, 使用同样的公式

$$x_{k+1} = x_k - \frac{x_k^3 + x_k - 1}{3x_k^2 + 1} = \frac{2x_k^3 + 1}{3x_k^2 + 1},$$

迭代 6 次. 观察并比较计算结果, 分析原因.

3. 几种经典算法的比较

实验题目：求方程 $f(x) = x^3 - \cos x - 5x - 1 = 0$ 的全部根.

方案一 用牛顿法求解：

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} = x_k - \frac{x_k^3 - \cos x_k - 5x_k - 1}{3x_k^2 + \sin x_k - 5};$$

方案二 用简单迭代法求解：

$$x_{k+1} = \sqrt[3]{\cos x_k + 5x_k + 1};$$

方案三 用艾特肯迭代加速法求解：

$$y_k = \varphi(x_k), \quad z_k = \varphi(y_k),$$

$$x_{k+1} = x_k - \frac{(y_k - x_k)^2}{z_k - 2y_k + x_k}, \quad k = 0, 1, \cdots,$$

其中

$$\varphi(x) = \sqrt[3]{\cos x + 5x + 1}.$$

取相同的迭代初始值, 比较各方法的收敛速度.

实验七 矩阵特征值计算实验

1. 幂法的收敛性

实验题目：(1) 用幂法求下列矩阵的模最大特征值和相应的特征向量，精确到 6 位有效数字：

$$A=\begin{pmatrix}-2 & 1 & -2\\ 9 & -2 & 7\\ 4 & -1 & 3\end{pmatrix}.$$

(2) 用反幂法求下列矩阵的模最小特征值和相应的特征向量，精确到 7 位有效数字：

$$A=\begin{pmatrix}6 & 2 & 1\\ 2 & 3 & 1\\ 1 & 1 & 1\end{pmatrix}.$$

2. 求矩阵的特征值和特征向量

实验题目：分别用幂法、反幂法、QR 方法求下列矩阵的主特征值、模最小特征值、全部特征值：

$$A=\begin{pmatrix}4 & -1 & -2\\ -1 & 5 & 3\\ -2 & 3 & 7\end{pmatrix},\quad B=\begin{pmatrix}2 & 3 & 2 & 3\\ 3 & 3 & 2 & -1\\ 2 & 2 & 4 & 4\\ 3 & -1 & 4 & 4\end{pmatrix},$$

$$C=\begin{pmatrix}4 & -1 & 0 & 0 & 0 & 0\\ -1 & 4 & -1 & 0 & 0 & 0\\ 0 & -1 & 4 & -1 & 0 & 0\\ 0 & 0 & -1 & 4 & -1 & 0\\ 0 & 0 & 0 & -1 & 4 & -1\\ 0 & 0 & 0 & 0 & -1 & 4\end{pmatrix},\quad D=\begin{pmatrix}0 & 9 & 0 & 0 & 0 & 0\\ 5 & 0 & 8 & 0 & 0 & 0\\ 0 & 5 & 0 & 7 & 0 & 0\\ 0 & 0 & 5 & 0 & 6 & 0\\ 0 & 0 & 0 & 5 & 0 & 5\\ 0 & 0 & 0 & 0 & 5 & 0\end{pmatrix}.$$

附录二 MATLAB 软件入门

MATLAB 是美国 MathWorks 公司开发的一种集数值计算、符号运算、可视化建模和图形处理等多种功能于一体的非常优秀的图形化语言. 为了配合本教材的教学和学习, 本附录将对 MATLAB 语言及其程序设计基础进行简要的介绍, 包括 MATLAB 语言的基本要素、MATLAB 语言的核心 —— 矩阵、流程控制语句和 M 文件.

B.1 MATLAB 数值处理简介

MATLAB 是 Matrix Laboratory (矩阵实验室) 的英文缩写, 它的基本数据单元是不需要指定维数的矩阵. MATLAB 中矩阵变量名必须是以字母开头的、由字母和数字组成的字符串.

数值计算功能是数学软件 MATLAB 的基础和特色, 强大的数值计算功能使得 MATLAB 在诸多的数学软件中傲视群雄. 自商用的 MATLAB 软件推出之后, 它的数值计算功能就在不断地改善并日趋完善. 目前的 MATLAB R2007 版本更是把其数值计算功能推向了一个新的高度. 正是由于 MATLAB 有了如此令人惊叹的强大数值计算功能, Mathworks 公司才有能力把 MATLAB 的应用延伸到不同专业、不同行业和部门的各个领域, 使其成为世界上最优秀、应用最广泛、最受用户喜爱的数学软件.

B.1.1 向量及其运算

矩阵是 MATLAB 语言的基本数据单元, 向量可视为矩阵的特殊形式. 本小节对向量的生成及其基本运算做简单介绍.

1. 向量的生成

1) 直接输入向量

生成向量最直接的方法是在命令窗口中直接输入. 在格式上要求是: 向量元素需要用 "[ ]" 括起来, 元素之间可以用空格、逗号 (,) 或分号 (;) 分隔. 需要注意的是, 用空格或逗号分隔生成行向量, 用分号分隔生成列向量. 例如:

```
>> a=[1 3 5 7 9 11]
a =
    1    3    5    7    9    11
```

2) 利用冒号表达式生成向量

冒号表达式的基本形式是:

x=x1:step:xn,

其中 x1 表示向量的第一个元素的数值, xn 表示向量尾元素的数值限, step 表示前后两个元素的差值. 用冒号表达式生成向量可以不用“[ ]”. 例如:

```
>> b=1:2:12
b =
   1    3    5    7    9    11
```

注 B.1　(1) xn 为向量尾元素的数值限, 而非尾元素, 如上面的例子.

(2) 若 x0<xn, 则 step>0; 若 x0>xn, 则 step<0. 例如:

```
>> c=12:-2:1
c =
   12   10    8    6    4    2
```

(3) 若 step=1, 则可略去此项的输入, 直接写成 x=x0:xn. 例如:

```
>> d=1:8
d =
   1    2    3    4    5    6    7    8
```

3) 线性等分向量的生成

MATLAB 提供了线性等分功能函数 linspace, 用来生成线性等分向量, 其调用格式是:

```
y=linspace(x1,x2)   %生成100维的行向量, 使得y(1)=x1, y(100)=x2;
y=linspace(x1,x2,n) %生成n维的行向量, 使得y(1)=x1, y(n)=x2.
```

例如:

```
>> a1=linspace(1,10,5)
a1 =
   1.0000    3.2500    5.5000    7.7500   10.0000
```

4) 对数等分向量的生成

在自动控制、数字信号处理中, 常常需要对数刻度坐标. MATLAB 还提供了对数等分向量功能函数, 具体格式如下:

y=logspace(x1,x2)　%生成50维对数等分向量, 使得 $y(1)=10^{x1}, y(50)=10^{x2}$;

y=logspace(x1,x2,n)　%生成n维对数等分向量, 使得 $y(1)=10^{x1}, y(n)=10^{x2}$;

例如:

```
>> a2=logspace(0,4,5)
a2 =
   1      10      100      1000      10000
```

2. 向量的基本运算

1) 向量加 (减) 法

两个向量相加 (减) 必须维数相同, 例如:

```
>> b-c     % 这里的b,c即上面生成的b,c
ans =
   -11    -7    -3     1     5     9
```

2) 数与向量相加 (减)

数与向量相加是指数与向量的每个元素相加, 例如:

```
>> a1+3
ans =
    4.0000    6.2500    8.5000   10.7500   13.0000
```

3) 数与向量相乘

数与向量相乘是指数与向量的每个元素相乘, 例如:

```
>> a1*3
ans =
    3.0000    9.7500   16.5000   23.2500   30.0000
```

B.1.2 矩阵及其运算

MATLAB 的原意为矩阵实验室, 其所有的数值功能都是以矩阵为基本单元进行的, 因此, MATLAB 的矩阵运算功能可以说是最全面、最强大的. 本小节将对矩阵及其运算简要的阐述.

1. 矩阵的生成

1) 直接输入小矩阵

生成矩阵最方便、最常用的方法是从键盘上直接输入, 这种方法尤其适合较小的矩阵. 在用此方法创建矩阵时, 需要注意几点:

(1) 输入矩阵时要以“[ ]”为标识, 即矩阵的元素应在“[ ]”的内部;

(2) 矩阵的同行元素之间用空格或“,”号分隔, 行与行之间用“;”号或回车符分隔;

(3) 矩阵大小可不预先定义, 无任何元素的空矩阵也是合法的;

(4) 若不想获得中间结果, 可以用“;”号结束;

(5) 矩阵元素可以是运算表达式.

例如, 创建一个简单的数值矩阵:

```
>> A=[1 2 3;4 5 6;7,8,0]
A =
```

```
  1     2     3
  4     5     6
  7     8     0
```

又如，创建一个带有运算表达式的矩阵：

```
>> B=[sin(pi/4) cos(pi/3); tanh(4) log(6)];
```

此时已经建立并存储在内存中只是没有在屏幕上显示而已. 若想查看此矩阵，只需键入矩阵名.

2) 创建 M 文件输入大矩阵

M 文件是一种可以在 MATLAB 环境下运行的文本文件. 它分为命令式和函数式两种. 此处主要介绍用命令式的 M 文件来创建大型矩阵，更详细的内容将在后面介绍.

当矩阵较大时，直接输入就显得比较笨拙，且容易出错. 为解决此问题，可利用 M 文件的特点将所要输入的矩阵按格式先写入一个文本文件中，并将此文件存为以 m 为扩展名，即为 M 文件. 在 MATLAB 的命令窗口键入此 M 文件名，则所要输入的大型矩阵就被输入到内存中. 例如：编制一名为 a01.m 的 M 文件，内容如下：

```
%a01.m
%创建一M文件输入矩阵的示例
exm=[1 2 3 70 80 90;
    400 500 600 9 8 7;
    21 32 43 54 65 76;
    789 678 567 456 345 0;
    12 34 45 67 89 90];
```

在命令窗口键入：

```
>>a01
>> size(exm)
ans =
   5   6
```

说明：(1) 在 M 文件中，% 符号后面的内容只起注释作用，将不被执行；

(2) 函数 size 的功能是求矩阵的维数；

(3) 在通常的使用中，上例中的矩阵算不上“大型”矩阵，此处只是借此作一说明而已.

2. 矩阵的基本运算

矩阵的基本数学运算包括矩阵的四则运算，数乘矩阵运算，矩阵求逆等.

1) 矩阵的四则运算

矩阵的加减法：

矩阵的加减法使用“+”和“-”运算符, 格式于数字运算完全相同, 但要求两个矩阵是同阶的. 例如：

```
>> A=[1 2 3;4 5 6;7 8 9];
>> B=[2 2 2;3 3 3;1 1 1];
>> C=A+B
C =

    3    4    5
    7    8    9
    8    9   10
```

矩阵的乘法：

矩阵的乘法是用运算符“*”, 要求第一个矩阵列数与第二个矩阵的行数相同才能相乘. 例如：

```
>> D=[B,[4 4 4]']
D =

    2    2    2    4
    3    3    3    4
    1    1    1    4
>> E=A*D
E =

   11   11   11   24
   29   29   29   60
   47   47   47   96
```

矩阵的除法：

矩阵的除法可以有两种形式：左除“\”和右除“/”. 在低版本的 MATLAB 中, 右除是要先计算矩阵的逆再做矩阵的乘法, 而左除则不需要计算矩阵的逆而直接做除运算. 通常左除要快一点, 但在 MATLAB6.x 及其以上版本中两者的区别不大.

通常用矩阵除法来求解线性方程组. 对于方程组 $Ax=b$, 其中 A 是一个 $m\times n$ 矩阵, 则

(1) 当 $m=n$ 且 A 非奇异时, 此方程称为恰定方程；

(2) 当 $m>n$ 时, 此方程称为超定方程；

(3) 当 $m<n$ 时, 此方程称为欠定方程.

这三种方程都可以用矩阵的除法求解. 例如：

```
>> A=[3 2 1; 4 5 2;5 6 7];
>> b=[6 7 8]';
>> x=A\b
x =
    2.3125
   -0.3750
   -0.1875
```

2) 矩阵与常数间的运算

矩阵与常数间的运算是指矩阵的每一个元素与该常数做运算. 如数加是指每个元素都加上此常数, 数乘是指每个元素都乘以此常数. 需要注意的是, 当进行数除时, 常数通常只能做除数. 例如:

```
>> A=[1 2 3;3 4 5;5 6 7];
>> B=A+5
B =
     6     7     8
     8     9    10
    10    11    12
>> C=5*A
C =
     5    10    15
    15    20    25
    25    30    35
>> D=A/5
D =
    0.2000    0.4000    0.6000
    0.6000    0.8000    1.0000
    1.0000    1.2000    1.4000
```

3) 矩阵求逆

矩阵的求逆运算是矩阵运算中一种重要的运算. 它在线性代数和数值分析中都有很多的论述, 而在 MATLAB 中, 众多的复杂理论归结为一个简单的命令 inv. 例如, 求下列矩阵的逆:

```
>> A=[4 3 2 1;1 4 3 2;5 6 7 4;6 7 8 9];
>> B=inv(A)
B =
    0.2750   -0.2500         0    0.0250
```

```
    0.2250    0.5000   -0.2500   -0.0250
   -0.3750   -0.2500    0.5000   -0.1250
   -0.0250    0.0000   -0.2500    0.2250
>> A*B
ans =
    1.0000   -0.0000         0   -0.0000
    0.0000    1.0000         0   -0.0000
    0.0000    0.0000    1.0000   -0.0000
    0.0000    0.0000         0    1.0000
```

3. 特殊矩阵的生成

1) 空阵

在 MATLAB 中定义“[ ]”为空阵. 一个被赋予空阵的变量具有如下性质:

(1) 在 MATLAB 工作内存中确实存在被赋空阵的变量;

(2) 空阵中不包含任何元素, 它的阶数是 0×0;

(3) 空阵可以参与各种矩阵运算.

2) 几种常用的工具阵

以下各种常用的工具阵, 除了单位阵外, 其它的似乎没有任何具体意义, 但它们在实际中有着广泛的应用. 例如定义矩阵的维数和给迭代矩阵赋初值等. 这些工具矩阵主要包括全 0 阵、单位阵、全 1 阵和随机阵等.

(1) 全 0 阵: 全 0 阵可由函数 zeros 生成, 其主要调用格式为:

(a) zeros(N) % 生成 N×N 阶的全 0 阵;

(b) zeros(M, N) % 生成 M×N 阶的全 0 阵;

(c) zeros(size(A)) % 生成与 A 同阶的全 0 阵.

(2) 单位阵: 单位阵可由函数 eye 生成, 其主要调用格式为:

(a) eye(N) % 生成 N×N 阶的单位阵;

(b) eye(M, N) % 生成 M×N 阶的单位阵;

(c) eye(size(A)) % 生成与 A 同阶的单位阵.

(3) 全 1 阵: 全 1 阵可由函数 ones 生成, 其主要调用格式为:

(a) ones(N) % 生成 N×N 阶的全 1 阵;

(b) ones(M, N) % 生成 M×N 阶的全 1 阵;

(c) ones(size(A)) % 生成与 A 同阶的全 1 阵.

(4) 随机阵: 随机阵可由函数 rand 生成, 其主要调用格式为:

(a) rand(N) % 产生一个 N×N 阶均匀分布的随机矩阵;

(b) rand(M, N) % 产生一个 M×N 阶均匀分布的随机矩阵;

(c) rand(size(A)) % 产生一个与 A 同阶均匀分布的随机矩阵；

(d) randn(N) % 产生一个 N×N 阶正态分布 (N(0,1)) 的随机矩阵；

(e) randn(M, N) % 产生一个 M×N 阶正态分布的随机矩阵；

(f) randn(size(A)) % 产生一个与 A 同阶正态分布的随机矩阵.

B.2 MATLAB 程序设计入门

MATLAB 作为一种应用最广泛的科学计算软件, 它不仅具有强大的数值计算、符号运算等功能, 而且可以像 Basic、C、Fortran 等计算机高级语言一样, 进行程序设计、编写 M 文件. 通常, MATLAB 被称为第四代编程语言, 从而也足见其简洁. 编写 M 文件与编写 Basic、C、Fortran 等计算机高级语言程序相比, 有其独特的、无法比拟的优点, 如语言结构简单、程序语句可读性好、程序调试简便易行等. MATLAB 提供的 M 文件编辑器可以大大方便用户进行各种各样的程序设计.

B.2.1 运算符和操作符

MATLAB 的运算符可以分为下面三类: 算术运算符、关系运算符和逻辑运算符. 现将 MATLAB 的三大类别的运算符简单介绍如下.

1) 算术运算符

算术运算符是构成运算的最基本的操作命令, 可以在 MATLAB 的命令窗口中直接运行.

符号	意义	符号	意义
+	相加	-	相减
*	矩阵相乘	.*	数组相乘
\	矩阵左除	.\	数组左除
/	矩阵右除	./	数组右除

例如, 比较下面的计算结果, 体会运算符"*"与".*", "\"与".\"的区别.

```
>> A=[3 2 1; 5 6 4; 7 8 9];
>> B=[1 1 1; 2 2 2; 3 3 3];
>> C=A*B
C =
    10    10    10
    29    29    29
    50    50    50
>> D=A.*B
D =
```

```
    3     2     1
   10    12     8
   21    24    27
>> E=A\B
E =
    0.2667    0.2667    0.2667
    0.0667    0.0667    0.0667
    0.0667    0.0667    0.0667
>> F=A.\B
F =
    0.3333    0.5000    1.0000
    0.4000    0.3333    0.5000
    0.4286    0.3750    0.3333
```

2) 关系运算符

关系运算符主要用来比较数、字符串、矩阵之间的大小或不等关系，其返回值是 0 或 1. 关系运算符主要有

符号	意义	符号	意义
>	大于	<	小于
>=	大于等于	<=	小于等于
==	等于	=	不等于

例如：

```
>> a=[1 2 -3; 4 -5 6; 7 8 -9]
a =
    1    2   -3
    4   -5    6
    7    8   -9
>> b=a>1
b =
    0    1    0
    1    0    1
    1    1    0
```

3) 逻辑运算符 MATLAB 语言有四种基本的逻辑运算：&(与), |(或), ~(非) 和 xor (异或). 逻辑表达式和逻辑函数的值应该是一个逻辑量“真”或“假”，其中以 0 表示“假”，以任意非 0 数表示“真”. 例如：

```
>> a=[0 1 2 0];
>> b=[4 3 0 0];
>> a&b
ans =
     0     1     0     0
>> a|b
ans =
     1     1     1     0
>> xor(a,b)
ans =
     1     0     1     0
>> ~a
ans =
     1     0     0     1
```

B.2.2 M 文件简介

就实质而言, MATLAB 是一种解释性语言. 用 MATLAB 编程语言编写的可以在 MATLAB 工作空间运行的程序, 称为 M 文件. M 文件可以根据调用方式的不同分为命令文件和函数文件. 命令文件不需要任何输入参数, 也不返回任何输出参数, 它只是命令的简单叠加, MATLAB 会自动按照顺序执行文件中的各个语句行. 这样就解决了在命令窗口反复运行诸多命令的繁琐, 还可以避免用户做很多重复性的计算、分析等工作. 函数文件通常包含输入参数, 也可以返回输出参数, 它主要解决参数传递和函数调用的问题. 函数文件的第一个语句行必须以 function 语言为引导. 另外, 命令文件对工作空间中的变量进行操作, 而函数文件中的变量为局域变量, 只有其输入和输出的变量保存在工作空间中.

1. 命令文件

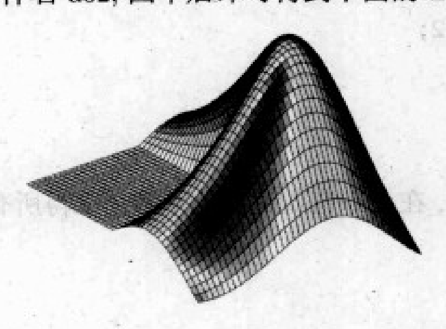
由于命令文件没有输入输出参数, 只是一些命令行的组合, 它的运行相当于在命令窗口逐行输入并运行命令. 因此用户在编写此类文件时, 只需把想要执行的命令按行编写到指定的文件中, 而且变量不需要预定义, 也不存在文件名的对应问题. 但是在编写命令文件时要注意标点符号的运用, 最后不要忘记 M 文件的扩展名.

实例: 建立一个 M 文件来绘制 MATLAB 的 LOGO 图标. 编制命令文件如下:

```
%a02.m
load logo          %装载LOGO数据
surf(L,R)          %根据数据绘制LOGO图标
```

```
n=size(L,1)        %获取数据矩阵维数
axis off           %关闭坐标轴
axis([1 n 1 n -0.4 0.5])   %设置图形显示的区域
view(-37.5,40)   %设置视点
```

在命令窗口键入文件名 a02, 回车后即可得到下面的 LOGO 图标：



2. 函数文件

1) 函数文件的组成

与命令文件相比, 函数文件要复杂一些. 通常, 要实现计算中的参数传递和函数的嵌套调用等, 需要编写函数文件. 函数文件可以有返回值, 也可以只执行而没有返回值. 函数文件在 MATLAB 中应用最广泛, MATLAB 提供的绝大多数功能函数都是由函数文件实现的. 函数文件通常包括以下几部分：

a. 函数的定义行. 文件的第一行为函数的定义行, 该行定义函数名、输入变量和输出变量；

b. 函数帮助信息行. 用于给出函数的帮助信息, 注意每一行需要以 % 开始；

c. 函数体. 函数的功能实现部分, 它是完成函数编写的主要部分；

d. 注释部分. 在函数体中以符号 % 开始直到该行结束的部分表示对程序的注释.

函数文件执行后, 只保留最后的返回结果, 不保留任何中间过程, 所定义的变量也都是局域变量, 仅对函数内部起作用, 并且随着函数调用的结束而从工作空间清除.

2) 函数文件的生成过程

用 M 文件编辑器编写函数文件如下：

```
%a03.m
function f=a03(n)
%求小于某自然数且为2的整数次幂的正整数
%调用格式: c=a03(n)
```

```
%参数说明: n可以取任意正整数
% 2006年8月编制
f(1)=2;
    i=1;
    while f(i)<ceil(n/2)
    f(i+1)=f(i)*2;
    i=i+1;
end
f;  %返回值
```

根据以上函数文件, 在命令窗口键入以下命令求得所有小于 10000 的 2 的正整数次幂.

```
>> c=a03(10000)
c =
     2      4      8     16     32     64
   128    256    512   1024   2048   4096
  8192
```

说明:

a. 第一行的作用是通过 function 声明了该文件是函数文件, 包括函数名的定义, 输入输出参数的定义;

b. 函数体内的变量 i 为局域变量, 当函数调用完成后, 该变量被释放;

c. 函数文件的前几行为注释行, 这些注释可以通过 help 命令查询.

B.2.3 流程控制语句

MATLAB 的程序结构一般可分为顺序结构、循环结构和分支结构三种. 顺序结构是指程序语句按顺序逐条执行, 循环结构和分支结构都有其特定的语句, 这样可以增强程序的可读性. 这里主要介绍两种循环结构 (for···end 与 while···end) 和两种分支结构 (if···else···end 与 switch···case···end).

1. FOR 循环结构

这个结构用于循环执行某些语句, 每执行一次就要判断是否继续执行. 这个判断依据的条件就是循环终止条件. 这种循环的语法结构如下:

```
for i = 初值:增量:终值
  语句1;
  ......
  语句n
```

```
end
```

说明：for 与 end 之间的语句成为循环体. i 是循环变量, 初值、增量和终值可正可负, 可以是整数, 也可以是小数, 只要符合数学逻辑. 增量可以自己设定, 也可以缺省, 缺省值为 1. 这样, 变量 i 从初值开始, 循环体中的语句每被执行一遍 i 就增加一个增量, 直到"i= 终值"为止.

例如：设计一个程序, 计算阶乘 10!

MATLAB 程序如下：

```
%a04.m
clear;
prod=1;
for i=1:10
    prod=prod*i;
end
prod
```

在命令窗口键入 a04, 回车得

```
>> a04
prod =
     3628800
```

2. WHLIE 循环结构

While 循环结构也使用循环执行某些语句, 但是这种结构与 for 循环不同之处在于, 在执行循环体之前先判断循环执行的条件是否成立, 如果成立则执行, 否则终止循环. 这种循环结构的语法结构如下：

```
while 逻辑表达式
    循环体语句
end
```

说明：(1) While 结构依据逻辑表达式的值决定是否执行循环体语句. 若逻辑表达式的值为真, 则执行循环体语句一次, 然后返回再次判断逻辑表达式的值是否为真. 在反复执行时, 每次都要判断. 若逻辑表达式的值为假, 则程序转到执行 end 之后的语句;

(2) While 循环内部还可以嵌套循环结构, 其形式如下：

```
while 逻辑表达式1
  循环体语句1
    while 逻辑表达式2
      循环体语句2
```

```
    end
    循环体语句3
end
```

例如：设计一个程序，求 1 至 100 之间的所有偶数的和.

MATLAB 程序如下：

```
%a05.m
clear;
x=2; sum=0;
while x<=100
    sum=sum+x;
    x=x+2;
end
sum
```

在命令窗口键入 a05，回车得

```
>> a05
sum =
    2550
```

3. IF 分支结构

If 结构是一种条件分支结构，判断某个条件是否成立，如果成立则执行结构内的语句，否则就跳出 if 分支结构，执行后面的语句. 这个分支结构的具体格式如下：

```
if 表达式1
    语句1
else if 表达式2 (可选)
        语句2
    else(可选)
        语句3
    end
end
```

说明：If 结构首先判断“表达式 1”是否成立，如果“表达式 1”成立，就执行“语句 1”，否则的话就执行 else if 中的条件判断. 如果结构中所有的表达式都不满足，就跳出执行本结构后面的语句. else if “表达式 2”与 else 为可选项，这两条指令可依据具体情况取舍. 注意，每个 if 都对应一个 end，即有几个 if，就应有几个 end.

例如：编程计算下列函数的值

$$f(x)=\begin{cases}\cos x, & x<0,\\ \sin x, & 0\leqslant x<1,\\ \ln x, & x\geqslant 1.\end{cases}$$

MATLAB 程序如下：

```
%a06.m
function y=a06(x)
if x<0
   y=cos(x);
else if x<1
       y=sin(x);
   else
       y=log(x);
   end
end
```

又如：输入一个正整数 n, 计算所有被 3 整除且小于 n 的正整数个数. MATLAB 程序如下：

```
%a07.m
n=input('Please input a number n=');
k=0;
for i=1:n
   if mod(i,3)==0
     k=k+1;
   end
end
k
```

在命令窗口运行该程序

```
>> a07
Please input a number n=1000
k =
   333
```

4. SWITCH 分支结构

Switch 结构是一种典型的分支结构, 根据表达式的结果执行后面跟表达式一致的 case 中的语句组. 这种分支结构可以表示比 if 结构更多的备选情况, 其具体形式如下：

```
switch 表达式
   case 常量表达式1
      语句组1
   case 常量表达式2
      语句组2
      ......
   otherwise
      语句组n
end
```

说明：

1. Switch 后面的表达式可以是任何类型, 如数字、字符串等;

2. 当表达式的值与 case 后面常量表达式的相等时, 就执行这个 case 后面的语句组. 如果所有的常量表达式的值都与这个表达式的值不等时, 则执行 otherwise 后面的语句组.

例如: 输入一个数, 判断它是否能被 3 整除. MATLAB 程序如下:

```
%a08.m
n=input('input a number n=');
switch mod(n,3)
   case 0
       fprintf('%d is a multiple of 3\n',n)
   otherwise
        fprintf('%d is not a multiple of 3\n',n)
end
```

在命令窗口运行该程序得

```
>> a08
input a number n=297
297 is a multiple of 3
>> a08
input a number n=389
389 is not a multiple of 3
```

5. 程序流程控制命令

1) Continue 命令

Continue 语句通常用于 for 循环或 while 循环体中, 其作用就是结束本次循环, 即跳过循环体中尚未执行的语句, 接着进行下一次循环.

例如：观察下面程序的运行结果并说明原因.

```
%a09.m
clear;
a=4; b=5;
for i=1:3
   b=b+1
   if i<2
      continue;
   end
   a=a+2
end
```

运行结果：

```
>> a09
b =
     6
b =
     7
a =
     6
b =
     8
a =
     8
```

程序说明：

a. 当 if 条件满足时，程序不再执行 continue 后面的语句，而是开始下一轮的循环. b. Continue 语句常用于循环体中，与 if 一同使用.

2) Break 命令

Break 语句也通常用于 for 循环或 while 循环体中，与 if 一同使用. 当 if 后的条件为真时，跳出当前循环. 通过使用 break 语句，可以不必等待循环的自然结束.

例如：鸡兔同笼问题. 某人数笼子里面的鸡和兔子的个数一共 36 只，腿数 100 只，问鸡兔各几何？MATLAB 程序如下：

```
%a10.m
function [x,y]=a10(g,t)  %g表示总个数, t表示总腿数
i=1;  %i表示鸡的个数
while i
```

```
        if rem(t-i*2,4)==0 & (i+(t-i*2)/4)==g
            break;
        end
        i=i+1;
    end
    x=i; y=(t-2*i)/4;
```

在 MATLAB 命令窗口调用该函数计算如下：

```
>> [x,y]=a10(36,100)
x =
    22
y =
    14
```

3) Return 命令

Return 命令能够使得当前的函数正常退出. 这个语句通常用于函数的末尾, 以正常结束函数的运行. 当然, 该函数也常用于其它地方, 首先对特定条件进行判断, 然后根据需要, 调用该语句终止当前运行, 并返回.

例如：输入两个正数, 求其中最大数. MATLAB 程序如下：

```
%a11.m  %主程序
clear;
a=input('Please input a number:');  %输入一个数
b=input('Please input a number:');  %输入另一个数
nummax(a,b);   %调用子函数
%
%nummax.m   %子函数程序
function nummax(a,b)
if a<=0 | b<=0
    disp('Input error')
    return;
    else if a>b
        fprintf('The larger number is %f\n',a);
    else
        fprintf('The larger number is %f\n',b);
    end
end
```

运行结果：

```
>> a11
Please input a number:8  %输入8
Please input a number:5  %输入5
The larger number is 8.000000  %运行结束
>> a11
Please input a number:7   %输入7
Please input a number:-4   %输入-4
Input error  %运行结束
```

B.3 MATLAB 绘图功能简介

计算结果的可视化是当今科学与工程计算的主导方向之一, MATLAB 提供了许多可以选用的图形功能. 本节简单介绍 MATLAB 的绘图功能.

B.3.1 二维图形函数

绘制二维图形最常用和最简单的绘图命令是 plot, 它的基本调用格式如下:

```
plot(x, y, '-');    % 向量 x 和 y 的维数必须相同
```

将向量 x 和 y 对应元素定义的点依次用实线连接起来; 如果 x 和 y 为矩阵, 则按列依次处理.

plot 命令的另一个调用格式是:

```
plot(x1, y1, '*', x2, y2, '+');
```

将向量 x1 和 y1 对应元素定义的点用星号标出, 将向量 x2 和 y2 对应元素定义的点用“+”标出. 即 MATLAB 可以画线或点, 它提供的点和线基本类型如下:

线型	符号	线型	符号	线型	符号
实线	-	虚线间点	-.	叉号	×
虚线	--	实心圆点	.	加号	+
点线	:	空心圆点	∘	星号	*

另一个二维图形函数是 fplot, 用于绘制已定义函数在指定范围内的图像, 它与“plot”命令类似, 差别在于 fplot 可以根据函数的性质自适应地选择取值点.

我们知道, 彩色显示在计算可视化中是十分重要的. 颜色在绘图中的使用方法与线形的控制方法一样, 常用的颜色有:

颜色	符号	颜色	符号	颜色	符号
红色	r	绿色	g	紫红色	m
蓝色	b	黑色	k	蓝绿色	c
黄色	y	白色	w		

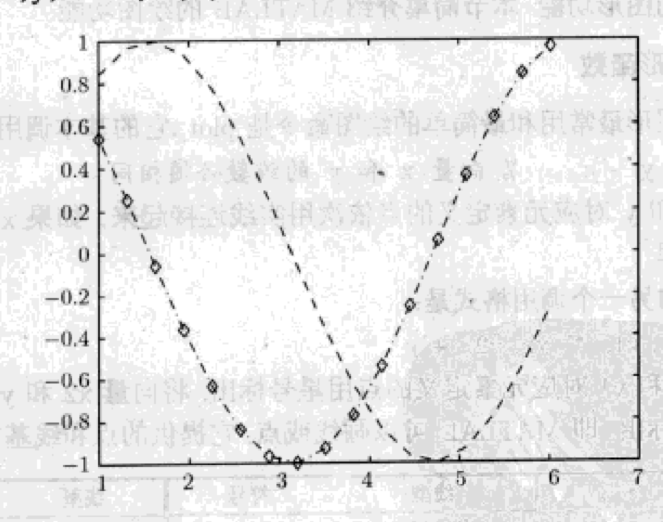
例如: plot(x,y,'-r') 表示将 x 和 y 对应元素定义的点依次用红色实线连接起来. 此外, 在绘图过程中还可以使用下面的线型:

线型	符号	线型	符号	线型	符号
方形	s	菱形	d	五角星	p
上三角	∧	下三角	v	左三角	<
右三角	>				

例如: 绘制下图所示带有显示属性设置的二维图形.

在 MATLAB 命令窗口依次执行下列语句:

```
>> x=1:0.1*pi:2*pi;
>> y=sin(x);
>> z=cos(x);
>> plot(x,y,'--k', x,z,'-.rd')
```



B.3.2　绘图辅助函数

利用下面的绘图辅助函数可以为画出的图形加上标题等内容.

函数	功能
grid	在图上显示虚线网格
hold on	后面 plot 的图形将叠加在一起
hold off	解除 hold on 命令, plot 将先冲去已有图形
title('...')	在图形的上方显示 ' ' 中所指定的内容
xlabel('...')	将 ' ' 中所指定的内容标在 x 轴上
ylabel('...')	将 ' ' 中所指定的内容标在 y 轴上
text(x, y,'...')	将 ' ' 中所指定的内容显示在 x, y 所定义的位置上
axis([xl, xr, yl, yr])	其中的 4 个数分别定义 x, y 方向的显示范围

注意上述辅助绘图命令必须放在相应的“plot”语句之后. 此外, 还有一个辅助命令：gtext('...'); 运行到该命令时, 屏幕光标位置显示符号“+”等待, 它将 '' 中所指定的内容标在鼠标所指的位置.

例如, 绘制 Chebeshev 多项式的图形. Chebeshev 多项式可以用递推公式定义如下：

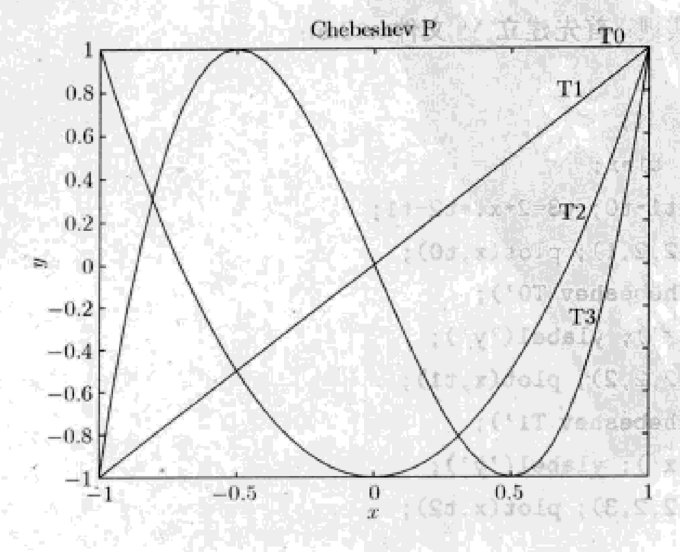
$$t_0(x) = 1, \quad t_1(x) = x,$$

$$t_n = 2x\,t_{n-1}(x) - t_{n-2}(x), \quad n = 2, 3, \cdots.$$

下面的程序将 $[-1,1]$ 上的前四个 Chebeshev 多项式画在一张图上.

```
>> x=-1:0.05:1;
>> t0=1.0;
>> t1=x;
>> t2=2*x.*t1-t0;
>> t3=2*x.*t2-t1;
>> plot(x,t0); gtext('T0');
>> title('Chebeshev P');
>> xlabel('x'); ylabel('y');
>> hold on
>> plot(x,t1); gtext('T1');
>> plot(x,t2); gtext('T2');
>> plot(x,t3); gtext('T3');
>> hold off
```

图形如下：

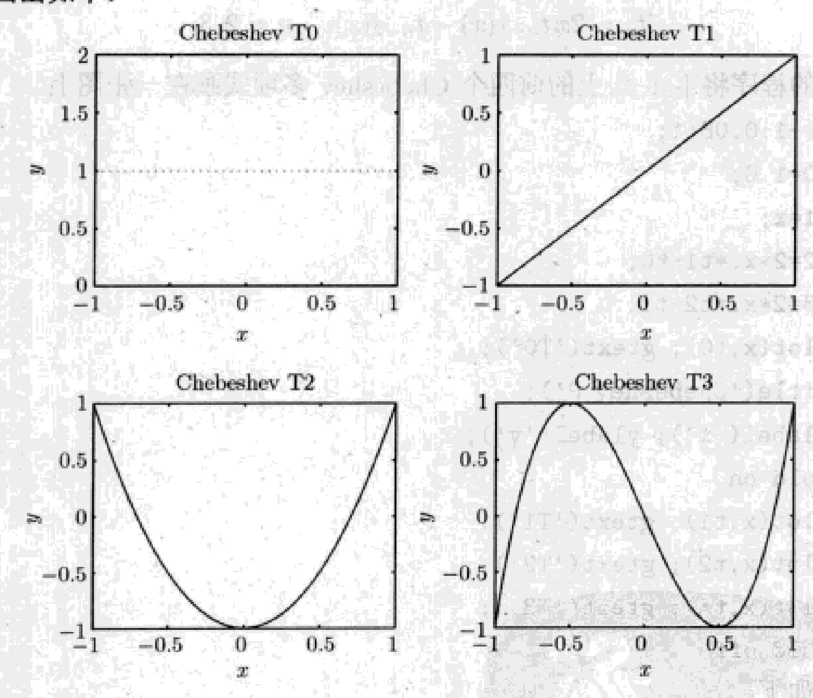


B.3.3 多窗口绘图函数

多窗口绘图函数为 subplot, 该函数的调用格式为：

subplot(m,n,r)

该命令将图形窗口分成 m 行 n 列共计 $m \times n$ 个格子, 在第 r 个上画图, 格子是从上到下按行一次记数的. 例如, 可以将前述的四个 Chebeshev 多项式以多窗口的形式画出如下：



可以这样实现：首先建立 M 文件 a12.m：

```
%a12.m
x=-1:0.05:1;
t0=1.0;  t1=x;
t2=2*x.*t1-t0; t3=2*x.*t2-t1;
subplot(2,2,1); plot(x,t0);
title('Chebeshev T0');
xlabel('x'); ylabel('y');
subplot(2,2,2); plot(x,t1);
title('Chebeshev T1');
xlabel('x'); ylabel('y');
subplot(2,2,3); plot(x,t2);
```

```
title('Chebeshev T2');
xlabel('x'); ylabel('y');
subplot(2,2,4); plot(x,t3);
title('Chebeshev T3');
xlabel('x'); ylabel('y');
```

然后在 MATLAB 命令窗口键入所建立的 M 文件名 a12, 回车后即得到所需要的图形.

B.3.4 三维图形函数

三维图形的绘图函数主要有 mesh 和 surf 两个, 其用法为:

mesh(x, y, z);　　%绘制网面图，其中 x，y，z 是同阶矩阵，表示曲面三维数据

surf(x, y, z);　%绘制网面图，与 mesh 用法类似

MATLAB 还提供了绘制等高线的函数 contour 和 contour3, 其用法为:

contour(x, y, z);　%绘制等高线图，与 mesh 用法类似

contour3(x, y, z);　%绘制三维等高线图，与 mesh 用法类似

此外, MATLAB 还提供了绘图辅助命令 meshgrid, 其功能和使用格式为

[x, y]=meshgrid(xa, ya);

当 xa, ya 分别为 m 维和 n 维向量, 得到 x 和 y 均为 n 行 m 列矩阵. meshgrid 常用来生成 x-y 平面上的网格数据.

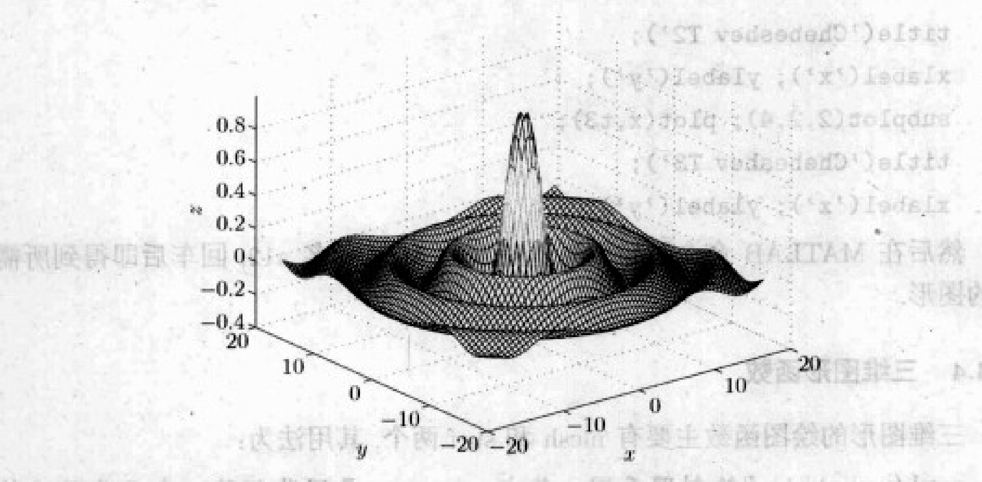
例如: 绘制函数

$$f(x,y)=\frac{\sin(\sqrt{x^2+y^2})}{\sqrt{x^2+y^2}}$$

的图像. 下面的程序画出该函数在区域 $(x,y)\in[-18,18]\times[-18,18]$ 上的三维图形.

```
%a13.m
clear;  close;
xa=-18:0.5:18;  ya=xa;
[x,y]=meshgrid(xa,ya);
z=sin(sqrt(x.^2+y.^2))./(sqrt(x.^2+y.^2)+eps);
mesh(x,y,z)
xlabel('x');  ylabel('y'); zlabel('z');
```

在 MATLAB 命令窗口执行文件 a13.m, 即得到下面的图形:



值得一提的是，可以使用科技排版软件 LaTeX 格式对 MATLAB 图形窗口中的图形进行诸如上标、下标、希腊字母等更为复杂的标注．例如，在标注内容中，可以使用 \bf,\it,\rm 黑体、斜体和正体；用 ^，_ 分别表示上标和下标；用 \alpha,\beta,\pi 等分别表示 α,β,π 等希腊字母；用 \leq,\geq,\times,\in, \cdot 分别表示 $\leqslant,\geqslant,\times,\in,\cdot$ 等数学符号．

例：绘制由参变量函数表示的空间曲线

$$\begin{cases} x=\mathrm{e}^{-0.2t}\cos\dfrac{\pi}{2}t, \\ y=\mathrm{e}^{-0.2t}\sin\dfrac{\pi}{2}t, \quad 0<t<20 \\ z=\sqrt{t}, \end{cases}$$

的图形．

下面的程序文件 a14.m 将一个图形窗口分成 1 行 2 列的两块，左边按默认的方式作图，并定义了标题和坐标轴；右边加了栅格，并将 z 轴范围限制在 $[0,4]$．

```
%a14.m
clear; close;
t=0:0.1:20; r=exp(-0.2*t); theta=0.5*pi*t;
x=r.*cos(theta); y=r.*sin(theta); z=sqrt(t);
subplot(1,2,1);
plot3(x,y,z);
title('螺旋线');
xlabel('X轴x=e^{-0.2t}cos(\pit/2)');
ylabel('Y轴'); zlabel('Z轴');
subplot(1,2,2); plot3(x,y,z);
```

```
axis([-1 1 -1 1 0 4]);
grid on;
```

在 MATLAB 命令窗口执行文件 a14.m, 即得到下面的图形: